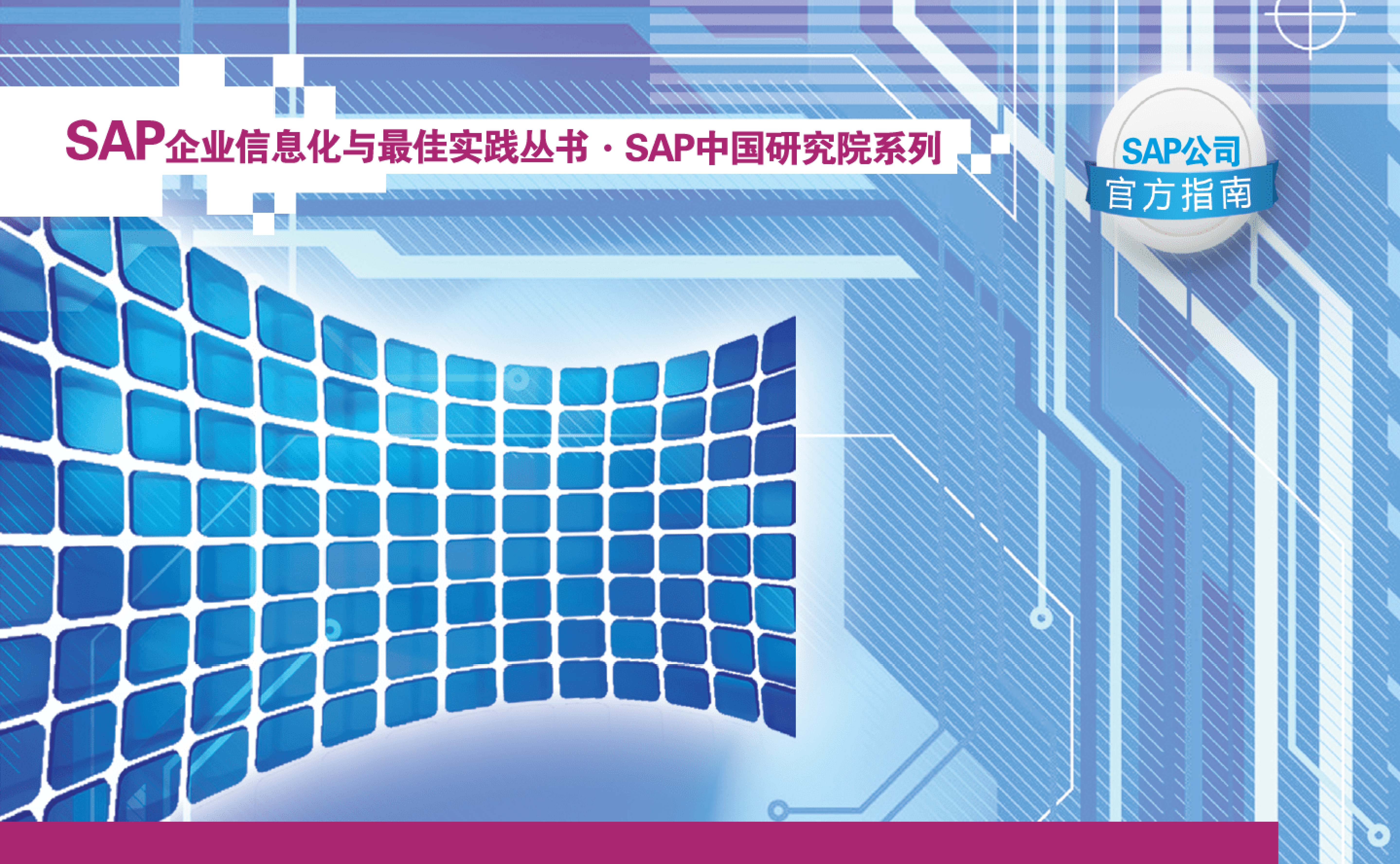


跟我学SAP HANA

——做大数据时代的领航者

尹东升　陈 磊　周 斌 / 著

清华大学出版社

SAP企业信息化与最佳实践丛书 • SAP中国研究院系列

跟我学 SAP HANA——做大数据时代的领航者

尹东升　陈磊　周斌　著

清华大学出版社

北　京

图书在版编目(CIP)数据

跟我学SAP HANA——做大数据时代的领航者 / 尹东升，陈磊，周斌 著. —北京：清华大学出版社，2014
(SAP企业信息化与最佳实践丛书•SAP中国研究院系列)

ISBN 978-7-302-35799-5

Ⅰ.①跟… Ⅱ.①尹… ②陈… ③周… Ⅲ.①企业管理—应用软件 Ⅳ.①F270.7

中国版本图书馆CIP数据核字(2014)第057791号

责任编辑： 陈 莉 蔡 琦
封面设计： 周晓亮
版式设计： 思创景点
责任校对： 邱晓玉
责任印制：
出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦A座　**邮　　编：** 100084
社 总 机： 010-62770175　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 175mm×226mm　**印　　张：** 26.25　**字　　数：** 469千字
版　　次： 2014年5月第1版　**印　　次：** 2014年5月第1次印刷
印　　数： 1～4000
定　　价： 48.00元

产品编号：

SAP企业信息化与最佳实践丛书·SAP中国研究院系列

编　委　会

Preface of the Series 1

More than ever, the ability to constantly innovate and re-invent ones business models and products makes the difference between those companies who are successful and those who disappear. The times are gone, that market shares could be cemented or protected by sheer dominance of a market or by classical marketing strategies. Local markets are progressively becoming merely a segment of the global market and every business, regardless of how local it may feel, has to face global competition. This is also true for China, where many companies have become global and are viable competitors for long established multinationals, even in their home markets. At the same time, with the opening of China as a market, most Chinese companies are facing tough competition through global companies too.

This competition can be fought through pricing, branding or access to distribution, but it will be finally decided who are the most flexible and innovative players. This is where SAP has a crucial role to play, as it enables companies through its most advanced software solutions, to react quickly to changes in demand, the market conditions or the customer behavior. Today, it is out most important to observe essential trends early enough – and at times we even need to predict certain changes in order to place significant investment and innovation resources on the right trends. Therefore world-class business intelligence tools are necessary; not only to analyze the past, but further to make the right prediction. Another important paradigm is speed. Decision must be made quickly and software needs to be able to analyze billions of data-sets within seconds to make business leaders having the right information in the right moment without delay. SAP HANA,



which is SAP's in-memory data base product, is the world's leading and fastest technology to read and analyze data. What we call "Big Data" is of course not only for "big companies" – every company is facing an enormous amount of internal and external information, that need to be processed. Consumer behavior is analyzed out of millions of statements in the internet, product reputation can be derived from microblogs or the DNA of a single person. Combined with the unique SAP business intelligence, SAP Business Objects, SAP is clearly a world market leader enabling its customers to innovate faster than ever before.

Obviously, the information needs to be available at the right place. On that account we talk about mobility and cloud. There is no doubt, that the way how we access and store data today has tremendously changed in the recent past. Today, China is the world's leading country when it comes to mobility, smart phone usage and internet access through mobile phones. Besides, in the area of cloud, China will make significant efforts to open this important market for global and local players alike.

Given those trends, it is important that every software engineer, every project manager in business and every student in the IT-field is updating him/herself constantly about the latest and most important trends and technologies in the fields of cloud, mobile, in-memory and business analytics. SAP is the company that is best positioned to provide insights not only into the latest technologies in these fields, but also business relevance that is uniquely combined with the technologies.

Practitioners from SAP Labs China are writing this book series and no-one than our own engineers, project managers and software architects would be in a better position to give deep insights into the latest technology trends, based on SAP's offerings in this context.

Enjoy the reading!

Senior Vice President, Head of SAP Labs Network,
Head of the Fast Growth Markets Organization
Clas Neumann

丛书总序 二

当今时代是信息技术飞跃发展的时代。2013 年，全球有超过 150 亿与互联网连接的设备，10 亿多人参与社交网络，每 18 个月，互联网上的数据会翻倍。以信息技术为中心的新科技革命，必然会推动经济运作形式发生巨大变化。信息技术广泛渗透到人们的生活和人类的生产各个环节并且快速地促进经济增长以及改变其增长模式。在信息时代，主要战略资源不再是工业时代原材料、土地、设备和能源，而是知识和信息。

IT 的发展尤其是云计算、大数据、移动化和社交化等技术新浪潮引发了一系列商业和社会变革，对企业信息化发展，企业经营和管理思维带来一系列深刻而且具有颠覆性的影响。跨界创新，产业链整合、产业生态体系竞争、信息技术与业务深度融合成为新时期 IT 产业发展的重要特点。而另一方面，伴随着中国新一轮经济体制改革的深化，整体的经济结构、发展战略和增长方式都在经历转型，使得企业亟需在运营模式、产品技术和市场战略等领域同步转型，以获取可持续的竞争优势。

作为众多变革性技术的领军者与推动者和 SAP 全球四大研究院之一，SAP 中国研究院及时顺应时代的潮流，推动战略转型，以 “Innovation in China for China and the World”(立足中国，创新中国，影响世界)为我们的战略方向，结合具体落地举措，在产品研发方面持续突破，并基于业界领先的 SAP HANA 平台，不断提升各个领域的研发工作，致力于打造面向不同行业、适合各类企业的产品和解决方案。

一个企业的成功由两个要素决定，一是与时俱进的战略方向，二是企业的核心竞争力。企业核心竞争力又包括三个基本要素：员工意愿、员工能力和员工治理。企业的成功，战略和核心竞争力二者缺一不可。在给定战略方向的情况下，如何提高员工的素质和能力，使员工具有活力和创新意识，关系到企业的可持续发展，对



企业的生存有着决定性的影响力。一个企业只有战略，而没有核心竞争力或卓越的团队执行力，那么战略只是一种美好的幻想；同样，一个企业没有与时俱进的战略方向，只是片面地强调团队执行力，那么这个企业只是行动主义者。由此可以看到，将战略方向和企业的核心竞争力进行有机结合才能把企业做大做强。因此，SAP 中国研究院在提出清晰的战略方向之后，推出了“零距离”的执行方案，提倡 4 个零距离：与客户零距离以促进创新，与生态体系零距离以共同成长，与内部各个部门零距离以保障知识的沉淀和传播，与员工零距离使管理层和员工、员工和员工沟通通畅，建立创新文化，既推动以客户为主的创新，同时推动驱动市场的创新。例如为了促进与生态圈的共同成长，SAP 中国研究院投入大量的资源为客户、合作伙伴、研发人员、大学生等在内的生态系统成员打造零距离的创新体验平台，例如 SAP d-code 技术大会，它是一场兼具实用性和前瞻性的技术盛宴，通过丰富扎实的课程内容和生动有效的互动模式，和生态圈共同提升，共同成长。同时，SAP 中国研究院致力于“SAP 企业信息化与最佳实践丛书 • SAP 中国研究院系列”的出版，把知识传播到生态圈。《跟我学 HANA——做大数据时代的领航者》就是执行战略转型 42 个项目计划之一。

在战略转型的过程中，SAP 中国研究院充分利用面向全球客户和中国客户的优势，汇集和融合全球客户最顶级的业务流程至软件之中，把我们的产品创新不断地送到客户手中，给我们客户带来附加价值，保障我们客户获取可持续的发展。同时，SAP 中国研究院更致力于深耕中国市场，向中国企业提供先进的管理解决方案，探讨企业如何挖掘技术力量、推进业务转型与可持续发展，帮助更多中国企业立足创新潮流，激发更大的商业潜能。这几年来，我们深耕企业应用、商务分析、移动商务、云计算、内存数据库及技术平台这五大市场领域，为各种行业、各种规模的企业研发了大量的创新解决方案。SAP HANA 产品是全新的内存数据库平台，作为一个驱动市场的创新产品，凭借其逻辑计算快、数据分析和抽取实时的优势吸引了国内外广大 SAP 产品的新老用户以及 SAP 合作伙伴的广泛关注。基于 HANA 的 SAP 云平台，支持全部 SAP 解决方案，推动企业在云端持续创新，从而促进企业的全面发展。

在推动 SAP HANA 内存数据库和平台的开发的同时，借助 SAP 内存数据库，SAP 对现有的企业管理软件如 ERP 进行了再造，提供一系列前所未有的新型企业应

用解决方案，其中结合了大量交易与实时分析能力，显著优化现有的业务流程、预测流程、定价优化流程等数据密集型流程。而基于 SAP 内存数据库的创新型业务分析功能可使我们的客户获得洞察力。同时，SAP 内存数据库平台和 SAP 业务分析产品的结合，又能使我们的客户对大数据进行快速挖掘，对企业经营进行前瞻性的预测和分析。

SAP HANA 的基本理念来自内存计算，可以说其是对行业影响深远的一款产品。作为一款灵活、多用途且与数据源无关的内存数据库兼平台，SAP HANA 是当代硬件和 SAP 软件的完美结晶。SAP 内存数据库采用改进的数据压缩、多栏式数据存储和内存计算技术，支持新一代企业数据管理。借助 SAP HANA，企业能够根据大量实时的详细信息分析，预测和以 Realtime 方式调整业务运营。随着互联网技术的发展和大数据时代的到来，它将能很好地满足大数据环境下对数据进行实时分析和处理的要求，这对企业来说有不可估量的价值。

在《跟我学 HANA——做大数据时代的领航者》中，SAP 中国研究院的同事以第一手资料，结合自己的开发经验而完成的，深入浅出地详细讲解了 SAP HANA 的技术和特点，并且理论和实践兼顾，希望读者能从中获得有关知识，从中受益。

SAP 正在中国拉开蓬勃发展的篇章，我相信清华大学出版社“SAP 企业信息化与最佳实践丛书 • SAP 中国研究院系列”的出版，将会让 IT 生态圈共同成长，让更多的有识人士深入了解最前沿的技术，与 SAP 共同创造新的辉煌！

最后，我要感谢对这套丛书的顺利出版提供帮助的各界人士，衷心感谢 SAP 中国研究院的同事完成这样一套高质量的丛书！

李瑞成博士

SAP 全球高级副总裁 SAP 中国研究院院长

2014年2月20日

序　言

自 2010 年公开面世以来，SAP HANA 正以 SAP 产品研发史上前所未有的速度完善和进化。随着越来越多的客户将核心业务迁移到 SAP HANA 上来，企业级内存计算技术已经被证明是高效可行的。基于 SAP HANA 搭建的技术平台和增值应用已经开始显著改变企业级计算环境，企业业务模式整体转变逐渐成为可能。我们欣喜地看到众多来自不同地域、不同行业的客户正从基于 SAP HANA 的解决方案中获益。

从应用开发人员的角度看，SAP HANA 凭借其广泛的标准兼容性降低了开发者学习的门槛。全面实现 ACID 意味着 SAP HANA 具备了事务型数据库的原子性、一致性、隔离性与持久性，对标准结构化查询语言以及 ODBC、JDBC、MDX 等常见接口的完整支持确保广大开发者可以直接将 SAP HANA 作为通用关系型数据库管理系统来使用。传统数据仓库开发人员稍加学习即可通过 SAP HANA 集成的可视化可编程多维数据建模工具建立复杂数据分析模型。基于内置的扩展应用服务，开发者可以使用熟悉的 C 语言、C++或者服务器端 JavaScript 语言轻松创建丰富的应用逻辑。其与 Hadoop、R 等开放环境的访问接口使得复用数据分析处理领域已有的研发成果成为可能。SAP HANA 的这些特性使广大不同专业技术背景的应用开发人员快速找到切入点，展开更深入的学习探索。

作为一项颠覆性的重大创新，SAP HANA 突破了传统企业级应用设计范式，使用单一系统同时处理联机事务型和分析型工作负载，将应用服务层合并到数据库管理系统内部，利用基于内存计算的列式存储技术提升数据处理性能并降低总体拥有成本，内嵌高效预测分析库以简化实时预测分析型应用开发，取消了默认数据表索引并原生支持文本分析及全文检索。企业级数据处理和分析应用通过 SAP HANA 这些创新的高级特性，其性能达到了极致。如何全面掌握各个组件的功能特点和使用方

法并加以综合运用，进而充分发挥SAP HANA平台的优势，对开发人员提出了更高的要求。

SAP 中国研究院的尹东升、陈磊、周斌三位同事结合近几年来在相关领域开发和实施项目中积累的丰富经验，将与客户和合作伙伴互动过程中经常被问到的各种问题加以归纳总结，编写了这本入门指南。读者不仅可以在书中找到搭建开发环境以及使用各主要开发工具的详细指南，而且可以了解 SAP HANA 主要组件的原理和应用场景。本书运用大量实例加以论述，数据翔实充分，阐述深入浅出，对有兴趣深入了解 SAP HANA 的开发人员具有非常有效的指导意义。

希望这本书能为广大开发人员打开一扇通向新一代企业级高性能海量数据处理应用开发之路的大门。

SAP中国研究院首席架构师

王　钊

2013 年 11 月于上海

前　　言

SAP HANA 问世已经有两年多了，相信大家通过各种渠道或多或少都了解了一些 SAP HANA 的主要特点，并对其前景有充分信心。与此相对应的是，大家可能从 SAP 官方网站得到大量的 SAP HANA 相关的英文知识点，但是用中文系统地涵盖 SAP HANA 各方面基础应用的书籍却极其罕见。编写此书的主要目的，就是把关于 SAP HANA 的基础知识用最简单易懂的形式呈献给大家，使得大家能够快速入门 SAP HANA 这个 SAP 公司推出的拳头产品，加快 SAP HANA 在中国的普及应用。书中的所有内容，都来源于我们自身对 SAP HANA 的深入了解以及在工作中参与的 SAP HANA 相关项目的经验总结。我们将这些内容整理成一个个知识点，在为每个知识点做讲解的同时，配以章节小结和练习来帮助大家巩固和掌握这些知识点。本书适合任何想进入 SAP HANA 世界又不知从何开始的读者，哪怕你以前并没有数据库与 SQL 基础，本书的内容也不会让你望而却步。如果你对 SAP HANA 已有一些了解，本书也是很好的实施指导手册，可以清晰地告诉读者在实施时应该做什么和不应该做什么。

为了照顾不同的读者群，我们将这本书分为两部分：第一部分主要面向想了解 SAP HANA 基础知识的读者，在这里我们首先介绍了 SAP HANA 所涉及的关于大数据、内存数据库以及内存计算相关的知识，这些基础知识是进入 SAP HANA 的敲门砖。随后我们进一步介绍 SAP HANA 的平台组成以及运用场景和系统构架，你将能从整体构架的高度来深入了解 SAP HANA 这个创新平台的组成以及特性。在第一部分的最后，你将会了解关于 SAP HANA 的一些实际应用案例以及 SAP HANA 相关实时资料的获取方式。

通过第一部分的阅读，你已经能从整体的角度对 SAP HANA 有比较详细的了解。但是如果你是偏技术性的读者或者想更加深入了解和运用 SAP HANA，那么你



可以通过第二部分的阅读来了解如何在 SAP HANA 中管理用户和权限，如何基于内存数据库创建列存储的数据表以及进行数据导入，如何使用工具来创建 SAP HANA 的数据模型，还能够在掌握这些知识点后利用 SAP HANA SQL 及存储过程进行数据开发和分析。这里需要着重指出的是，在第二部分的最后几章里，我们详细描述了诸如 OData、SAP HANA XS 以及 R 语言集成这些非常实用的内容，并提供了大量的可操作性案例来帮助你理解如何基于 SAP HANA 这个平台来创建基于 SAP HANA 的原生应用，你可以亲自体验使用 SAP HANA 提供的开发工具在短短的几分钟内创建这些原生应用的轻松又令人兴奋的操作过程。

可能有读者会问，SAP HANA 的服务器对于个人来讲非常昂贵，那我怎么能在阅读本书的同时完成相应的练习呢？在这里，我们贴心地为读者准备了一个章节，在这个章节中我们会告诉你怎么样申请一个基于“CloudShare”的免费的 30 天 SAP HANA 实例，以及在这个实例免费期到期后如何操作。其实在本书的其他章节中，这样的例子比比皆是，我们在本书的编写过程中一直遵循读者至上的宗旨，尽我们最大的能力使读者能在轻松愉快的氛围内完成本书所有操作和练习工作。

最后，我们希望大家在读完本书后，可以通过本书介绍的 SAP Community Network 与我们进行沟通，同时把你们的问题或者本书的不足反映给我们。我们将会不定期对大家提出的典型问题进行解答，或者编写一些最新的心得体会和大家分享。

尹东升　陈　磊　周　斌

2013年11月于上海

致　谢

首先我们要感谢 SAP 中国研究院院长李瑞成博士。李博士在得知我们写书的创意后表示了极大的赞同和支持，并亲自在本书编写的初期和我们一起从构架的高度为本书的主要目标和内容提出了指导性意见，在本书初稿完成后，他亲自审稿并对本书的内容提出了很有价值的修改意见。

另外我们还要特别感谢 SAP 中国研究院首席构架师王钊。他在百忙中从头到尾仔细阅读了我们的手稿，并针对本书的技术细节和案例实现方式提出了不少建设性的意见和建议。我们从中获益匪浅，并针对这些意见和建议对本书的内容做了大量修正和改进工作。与此同时，我们要感谢我们的同事王天越，她在工作之余对于本书的文字以及排版做了大量细致的修改，并对本书中很多段落进行了补充和润色。

最后，感谢 SAP 中国研究院的全球创新应用设计研发团队在案例及练习部分给予我们的大力支持。我们还要感谢部门领导对于此书的大力支持。同时感谢我们的同事们，多年来能与这些同为 SAP 中国研究院的精英们共事，我们感到很开心，谢谢他们在平时工作中给予的支持与帮助，我们将与他们在研讨会或者项目合作中所积累的点点滴滴的知识也一并在书中为大家呈现。

目　　录

第一部分　SAP HANA 基础知识篇

第一章　认识 SAP HANA ······ 3
第一节　SAP HANA 源起 ······ 3
第二节　先进的平台，前所未有的体验 ······ 5
第三节　大数据 ······ 7
第四节　SAP HANA 内存数据库 ······ 12
第五节　SAP 内存计算 ······ 21
第六节　SAP HANA 能做什么 ······ 25

第二章　SAP HANA 基础进阶 ······ 27
第一节　SAP HANA 平台详解 ······ 27
第二节　SAP HANA 组件架构 ······ 29
第三节　SAP HANA 应用场景 ······ 31
第四节　SAP HANA 应用开发 ······ 33
第五节　SAP HANA 企业云 ······ 35

第三章　SAP HANA 成功案例及实时资料 ······ 39
第一节　SAP HANA 成功案例 ······ 39
第二节　获取 SAP HANA 的最新资料 ······ 41

第二部分　SAP HANA 实践篇

第四章　从实例开始 SAP HANA 之旅……47

第五章　配置 SAP HANA 开发环境……49
第一节　申请试用 SAP HANA……49
第二节　SAP HANA Studio 视图……64
本章小结与练习……78

第六章　SAP HANA 用户与权限管理……79
第一节　概述……79
第二节　SAP HANA 特权……80
第三节　SAP HANA 角色管理……86
第四节　SAP HANA 用户管理……87
第五节　_SYS_REPO 权限管理……88
第六节　常见任务……89
第七节　常见问题……102
本章小结与练习……104

第七章　SAP HANA 数据供应……107
第一节　简介……107
第二节　本地文件方式数据导入……108
第三节　CTL 方式数据导入……111
第四节　SLT 方式数据导入……112
第五节　ETL 方式数据导入……118
本章小结与练习……128

第八章　在 SAP HANA Studio 中建立数据模型……129
第一节　基础概念……129
第二节　SAP HANA Studio 之数据表操作……136
第三节　SAP HANA Studio 之包简介……149

第四节 创建属性视图 ······ 156
第五节 创建分析视图 ······ 164
第六节 创建计算视图 ······ 173
本章小结与练习 ······ 182

第九章 SAP HANA SQL 基础 ······ 183
第一节 SAP HANA SQL 简介 ······ 183
第二节 定义数据 DDL ······ 189
第三节 操作数据 DML ······ 197
第四节 数据控制语言 ······ 214
第五节 关于 NULL 值 ······ 216
本章小结与练习 ······ 216

第十章 SAP HANA SQL Script ······ 219
第一节 SQL Script 简介 ······ 219
第二节 SAP HANA 存储过程 ······ 225
第三节 SQL Script 技术要点 ······ 243
第四节 SQL Script 开发注意事项 ······ 257
第五节 通过 ABAP 调用存储过程 ······ 266
本章小结与练习 ······ 270

第十一章 SAP HANA OpenData 服务 ······ 277
第一节 SAP HANA 扩展应用服务 ······ 277
第二节 SAP HANA XS-OData 服务 ······ 279
第三节 创建复杂的 OData 服务 ······ 310
本章小结与练习 ······ 324

第十二章 编写 SAP HANA 服务器端应用 ······ 325
第一节 创建简单的 XSJS 应用 ······ 325
第二节 应用程序编程接口 ······ 341
第三节 SAP HANA XSJS 库 ······ 349

本章小结与练习 ······ 354

第十三章　R 语言在 SAP HANA 中的应用 ······ 357
第一节　R 语言初步接触 ······ 357
第二节　R 语言基础 ······ 363
第三节　R 语言与 SAP HANA 集成配置 ······ 370
第四节　R 应用场景——嵌入式 R 代码 ······ 375
第五节　示例 ······ 378
本章小结与练习 ······ 384

第十四章　基于网页环境的 SAP HANA 工具 ······ 385
第一节　SAP HANA 简化版编辑工具 ······ 385
第二节　其他工具概览 ······ 396
本章小结与练习 ······ 398

第一部分

SAP HANA 基础知识篇

第一章　认识 SAP HANA

2011 年，SAP 公司将 SAP HANA™ 作为成熟的产品和解决方案向全球进行推广，引起了数据库领域强烈轰动。时至今日，SAP HANA 以其突破性的实时分析能力，在全球范围内已经给诸多客户的业务运营方式带来根本性的变革，并成为 SAP 史上用户数量增长最快的产品之一。

在本章节中，我们将详细介绍关于 SAP HANA 这一 SAP 全新平台的基础知识，包括什么是 SAP HANA，SAP HANA 能做什么。之后的章节中我们还将提到 SAP HANA 的应用案例及如何得到最新的关于 SAP HANA 的官方资料等。

第一节　SAP HANA 源起

近十几年来，随着企业信息化的全面普及，越来越多的 IT 系统(如 ERP 系统、企业电子商务网站、HR 系统等)为满足企业的日常运营需要逐渐被应用。随之而来的问题就是，企业用来进行分析决策的数据大部分分散在这些系统中间，难以有效地采集。同时，为这些数据建立统一视图，也是一个非常繁琐而又费时的过程。这些原因导致目前绝大部分企业的决策制定都是基于过时数据之上的，而无法根据最新数据进行实时分析决策。那么，有没有一种方案能够把企业这些分散的数据快速高效地收集起来，并以之作为企业实时分析和制定决策的基础呢？

自 1970 年关系型数据库首次被提出，其四十多年来都是围绕着一个几乎一成不变的理念，那就是数据库的数据都是存储在硬盘上的，只有需要时系统才会把数据“抓”到内存中做运算。但是由于企业的不断发展，其存储的数据量已经从原来的 KB 级、GB 级迅速呈 TB 级增长，数据类型也不知扩大了多少倍。在这种情况下，关系型数据库能否承载这些海量的、结构异常复杂的信息环境？与此同时，硬盘的容量虽然不断扩大，但其数据传输速度却鲜有提升。当前，物理硬盘I/O瓶颈的存在使得传统数据库在处理大量数据，尤其是涉及大量数据的范围查询时，会消耗

大量的时间，导致效率十分低下。那么，有没有一种办法能够彻底消除硬盘的 I/O 瓶颈，实现在数据处理过程中数据与处理器之间迅捷的互动呢？

当你使用 Web 搜索引擎搜索一个关键字时，每次按下回车键，浏览器即刻(通常是亚秒级响应[①])会把与搜索关键字相关的条目列出来。可能第一次出现的结果列表很模糊，但是你可以基于这些结果立即在列表内展开第二次或第三次搜索，直到找到你想要的结果。搜索引擎的快速响应，保证了你有足够的耐心去寻找结果。而在企业应用里面，这种亚秒级的查询响应非常少见，特别是一些涉及大量数据的复杂查询，终端用户往往需要等待几十分钟或者几个小时才能得到结果。由于时间过于漫长，用户可能已经没有兴趣再做进一步查询来挖掘数据更深的价值或者已经转向别的工作。特别是随着现在基于企业应用的移动终端设备大量普及，用户在这些设备上等待结果超过一分钟都是无法忍受的。那么，有没有一种平台，能够帮助企业级应用在此平台上实现如同搜索引擎一样亚秒级的用户响应呢？

假如你是企业前端工具(报表、查询工具、数据分析工具、数据挖掘工具)的开发者，你是否已经厌烦了数据抽取、数据上载、数据清理、数据集成、数据建模、数据组织，以及建立查询和信息立方体等一系列把数据从传统数据库转移到数据仓库，然后再转移到分析系统，最终到达前端工具的复杂数据操作？那么，你有没有想过如果数据库帮助你完成所有和数据运算相关的工作，你的应用只需要把数据库运算的结果数据展现出来将会怎样？不容置疑，这将大大减轻你的工作量，开发时间也会相应大幅缩短。而企业将是最大的受益者，因为开发效率的提高会节约大量的企业应用开发成本。

到目前为止，我们罗列出了一些企业常见的令人头疼的问题，总结起来为：

- 如何快速高效地收集企业大量的分散在各处的运营数据来为企业的决策提供实时分析？
- 如何完全消除传统硬盘的 I/O 瓶颈使数据处理快速进行？

---

① 通常观察者对刺激做出简单响应的平均反应时间为 220 毫秒。其中一部分时间用于发现刺激，其余时间用于做出响应。识别需要的响应时间更长，因为它还需要理解和领悟。识别的平均响应时间为 384 毫秒。此外，识别响应时间会随着环境复杂程度的增加而增加。在比较复杂的环境中，550 至 750 毫秒时间内做出的响应可以被称为“思想速度般的响应”。经过专业培训、重复执行同一动作的用户的反应时间会更短，因此，较慢的系统响应时间对他们来说会显得更加漫长。系统响应时间超过人反应时间的部分被视为等待时间。此时，用户会转移注意力，这是一个无法用意识控制的过程。等待的时间越长，用户放弃手头任务的可能性就越大。

- 如何能让企业级应用达到和当今搜索引擎一样的亚秒级响应？
- 如何能让数据库应用的开发者只考虑应用的本身，而不用在意数据层面的复杂运算？

是的，上面任何一个问题都非常棘手并且难以解决。但我们要告诉你的是，SAP HANA，这个 SAP 公司在 2011 年发布的划时代的变革性解决方案，一举解决了上述所有的问题。不要惊讶，随着本书的深入，你会发现对于当今热门的“大数据”和“云计算”所提出的一些挑战，SAP HANA 也提供了非常好的应对方案。好了，闲话不说，让我们先来认识一下 SAP HANA 吧。

第二节　先进的平台，前所未有的体验

经常会有人问，SAP HANA 到底是什么？“它是 SAP 新发布的分析报表系统吗？”“它就是个内存数据库，对吧？”“我觉得它就是个数据仓库。”面对这些问题，我们颇觉得有点盲人摸象的感觉。也许这些提问者或多或少了解一点 SAP HANA 的知识，但是由于不够全面，他们常常以点代面，以偏概全。

让我们首先来了解下 SAP HANA 到底是什么。SAP HANA™，英文全称为 SAP High-Performance Analytic Appliance，即 SAP 高性能分析设备，是一个由硬件(通过 SAP 硬件合作伙伴提供)及整合了基于硬件优化的 SAP 软件的模块组成的，专注于实时大数据分析和应用的先进平台。

我们知道，信息是当今企业的生命线，企业需要实时了解业务的运营状态。但遗憾的是，由于企业运营的数据量通常非常大，传统的磁盘系统无法在合理的时间内处理完成请求的数据，最终迫使企业不得不削减从运营应用程序导入分析模型的数据量，导致决策者对运营状况的了解大大滞后于采集的相应数据。通过部署 SAP HANA，企业能够把企业当前的和历史的海量数据全部放到服务器的内存中进行存储以及处理，彻底消除了传统数据库中物理硬盘 I/O 延时这一最大的性能瓶颈，通过 SAP 出色的内存计算技术，企业级应用可以对企业所有运营数据进行实时预测分析，并将结果迅速展现给终端用户来制定下一步决策。不仅如此，SAP HANA 平台把事务数据处理(OLTP)、分析数据处理(OLAP)以及应用逻辑处理等功能集于一身。这一开创性的变革彻底消除了以往企业日常业务运行数据和决策分析数据需要分别处理的壁垒，应用开发者无需提前对原始数据进行数据建模和物化聚合，由此消除了程序开发



的复杂性，这使开发者可以以更直接和清晰的方式创建基于 SAP HANA 平台的应用，实现了数据库应用的快速开发。目前来讲，企业不仅可以通过单独购买 SAP 硬件合作伙伴提供的装有预配置软件的设备[①]在企业内部部署 SAP HANA，也可以通过 SAP 云服务(SAP HANA Enterprise Cloud)来帮助企业管理和运维 SAP HANA 系统。如此一来，企业可以以更低的总体拥有成本更快速地实现价值，同时还可以畅享由世界领先的企业应用软件供应商 SAP 所提供的灵活性和可靠性。

图 1-1 为完整的 SAP HANA 构架图。从图中我们可以看出 SAP HANA 主要由 SAP BusinessObjects 数据服务组件、实时数据同步模块、SAP HANA 内存数据库、SAP HANA Studio 组成。其中 SAP HANA 的内存数据库(SAP HANA In-Memory Database，IMDB)是其重要组成部分，包括内存数据库服务器(In-Memory Database Server)、建模工具和客户端工具(ODBO、JDBC、ODBC、SQLDBC 等)。SAP 内存计算引擎(Computing Engine)是其核心，负责解析并处理对大量数据的各类操作，支持 SQL 和 MDX 语句、SAP 和非 SAP 数据。在这个平台之上，用户可以构建数据仓库或数据集市、报表和仪表盘等，并可以通过 SAP HANA 平台的扩展服务来开发基于 SAP HANA 的新应用。

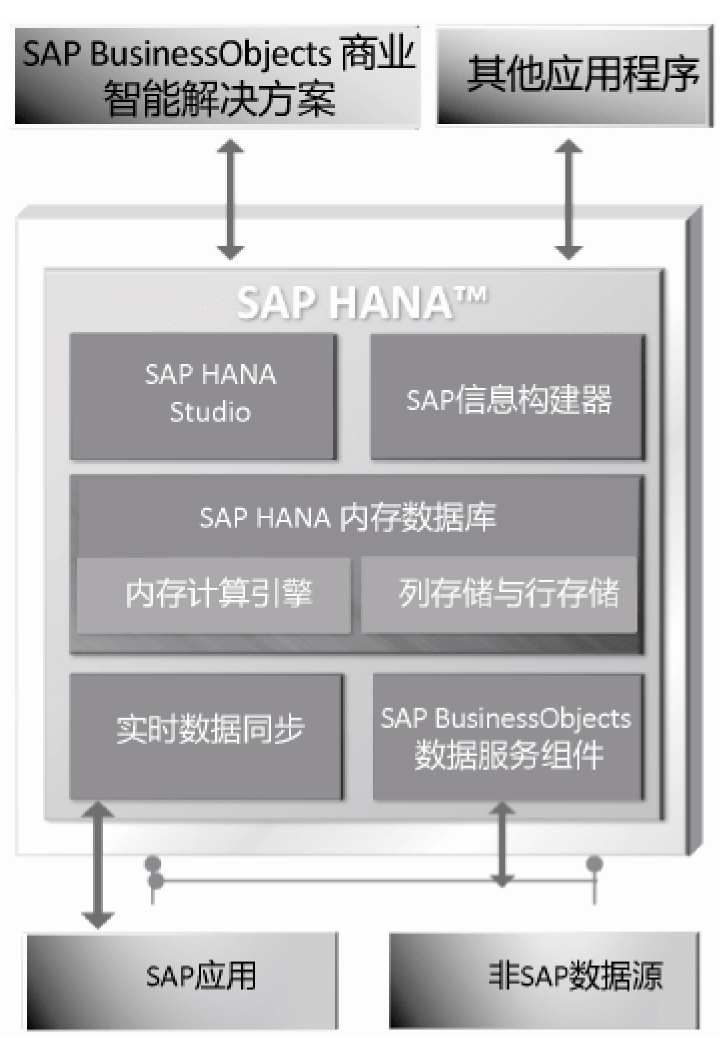


图 1-1



① SAP 有专门的 SAP HANA 硬件厂商认证计划，这个认证计划对全球的硬件厂商开放，目前，日立、IBM、惠普、戴尔等厂家都已经通过了认证，作为中国本土厂商的联想和华为也通过了 SAP 的这个认证计划，具体的厂商信息，请参照如下链接：https://service.sap.com/pam。

SAP HANA 的主要特点有：

- 软件＋硬件(HP，IBM，Cisco，Dell 等)的结合体
- 数据管理和数据建模
- 实时数据同步
- SAP BusinessObjects 数据服务组件不仅提供在 SAP 自身应用，如 SAP Business Suite、SAP NetWeaver Business Warehouse，同时也支持非 SAP 的第三方应用的数据采集—传输—上载功能。

SAP HANA 的能力主要有：

- 以史无前例的速度对海量非聚合数据进行实时分析
- 灵活的分析模式
- 多样的分析功能

好了，解释完 SAP HANA 的整体概念，接下来的小节里面将重点介绍 SAP HANA 的基本组成模块，如 SAP HANA 内存数据库、SAP 内存计算等。而在下一部分——SAP HANA 实践篇里面，会详细介绍更多的模块，例如数据同步、SAP HANA Studio 和 SAP HANA 平台的扩展应用。

在谈到基础模块之前，我们首先要了解一个概念，这不得不从 SAP HANA 出现的时代背景说起。从 2011 年开始，哪个词最热门，被人们谈论得最多呢？当然是“大数据”(big data)，而 SAP HANA 则是专门针对大数据做实时分析的先进平台。

第三节　大 数 据

百度每天执行超过 50 亿次搜索。

新浪微博有 5 亿注册用户，每天有 4650 万活跃用户并发布超过 10 亿条微博。

微信拥有超过 3 亿用户，每天传送数 10 亿条语音记录。

淘宝网站每天有超过数千万笔交易，单日数据产生量超过 50TB(1TB=1024GB)。

你有没有想过，上面这些令人惊讶的海量数据，其实就发生在我们每天的生活中。随着计算机和智能设备的普及，我们每天所看、所想、所做的事情都有可能转化为数据被记录下来并用于分析。以互联网为例，我们每天输入的搜索内容，访问电子商务网站所做的收藏、购买等操作以及论坛发帖等行为都会被记录并用于分



析。除了我们自身产生的数据，遍布全球的传感器、监控器、扫描仪等设备每天更是能产生海量的数据。从 2011 年开始，大数据一词越来越多地被提及，人们用它来描述和定义信息爆炸时代产生的海量数据，并命名与之相关的技术发展与创新。从分析的角度来讲，大数据具有以下特性。

一、申请 SCN 用户

我们处于人类历史上一个数据爆炸性增长的时代，我们从没有像现在一样产生如此巨量的数据。除去本小节开头那些互联网例子，再比如一个 8Mbps(兆比特每秒)的摄像头一小时能产生 3.6GB 数据，一个城市若安装几十万个交通和安防摄像头，每月产生的数据量将达几十 PB(1PB=1024TB)。医院也是数据产生集中的地方。现在，一个病人的 CT 影像数据量达几十GB，而全国每年门诊人数以数十亿计，并且他们的信息需要长时间保存。总之，大数据存在于各行各业，一个大数据时代已经到来。

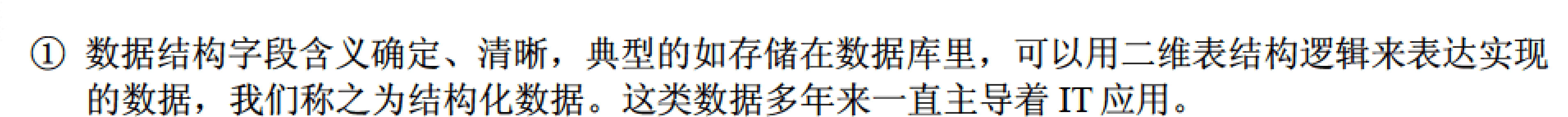
大数据的这一特性，导致单一数据集达到 TB 甚至 PB 级别已不罕见，进而使常规的数据抓取工具无法在允许的时间内对其进行捕获，大大提高了对其进行分析的难度。

二、类型多样(Variety)

大数据中包含的数据，不仅是传统的结构化数据①，还包含了大量的非结构化数据②。这些数据来自企业自身的业务数据，银行、股市的交易数据，更多的是来自图片、声音、视频、传感器信号、GPS 信息等广泛存在于社交网络、物联网、电子商务之中的数据。来自 IDC(国际数据公司)的研究报告指出，未来几年全球数据量将以 40%的速度增长，到 2020 年将达到 35ZB(35 万亿 GB)，其中 80%～90%为非结构化数据。

大数据的核心技术之一，就是要能够在这些复杂的数据类型中进行交叉分析，从而得到想要的结果。




① 数据结构字段含义确定、清晰，典型的如存储在数据库里，可以用二维表结构逻辑来表达实现的数据，我们称之为结构化数据。这类数据多年来一直主导着 IT 应用。

② 图片、声音、视频、传感器信号、GPS 信息等广泛存在于社交网络、物联网、电子商务之中的数据，我们称之为非结构化数据。

三、实时快速(Velocity)

正如前面讲到的，我们无时无刻不在生产数据，而且这些数据在爆炸式地增长，因此，基于这些数据的分析结果随时都可能发生变化，所以时效性对于大数据分析来说非常重要。通常我们使用的数据仓库系统和 BI 应用①对于时间的要求并不太高。人们对于一个大型报表运行 1～2 天才能出结果已经习以为常，但是对于大数据应用而言，必须要在极短的时间内形成答案，否则这些结果可能就是过时的、无效的。例如你想在早上出发前查询上班路线的实时路况，系统一个小时以后才把结果发送过来，那这个信息对你已经没有丝毫意义。

时效性要求高这一特性是大数据区别于传统数据挖掘最显著的特征，因此实时处理也成为许多提供大数据应用服务的机构需要面对的首要挑战。

综上所述，正是大数据的这些特性，决定了既有的传统技术架构和路线已经无法在合理的时间内高效处理如此海量的、种类繁多的数据。可以说，大数据时代对人类的数据驾驭能力提出了新的挑战，也为人们获得更为深刻、全面的洞察能力提供了前所未有的空间与潜力。因此，越来越多的政府、企业等机构开始意识到数据正在成为组织最重要的资产，数据分析能力正在成为组织的核心竞争力。

大数据所能产生的价值可以总结为如下 5 个方面：

- 先见之明：通过已经发生的、正在发生的事件或实验结果发现或预测需求，洞察变化倾向；
- 英明决策：自动算法代替/支持人类的决策；
- 一目了然：发现数据之间的关系；
- 有的放矢：细分人群，定制行动；
- 推陈出新：创新的商业模式、产品和服务。

或许，以下一些案例能帮你更快了解这些大数据的价值。比如，通过对社交媒

---

① BI 应用(BI-Business Intelligence)，即商业智能。商业智能的概念最早在 1996 年提出。当时商业智能被定义为一类由数据仓库(或数据集市)、查询报表、数据分析、数据挖掘、数据备份和恢复等部分组成的，以帮助企业决策为目的技术及其应用。目前，商业智能通常被理解为将企业中现有的数据转化为知识，帮助企业做出明智的业务经营决策的工具。商务智能系统中的数据来自企业其他业务系统，例如商贸型企业，其商务智能系统数据包括业务系统的订单、库存、交易账目、客户和供应商信息等；以及企业所处行业和竞争对手的数据、其他外部环境数据，这些数据可能来自企业的 CRM、SCM 等业务系统。



体数据、移动数据和网络数据等大数据的分析，企业可以充分了解自己的每一位客户，并结合客户的个性化特点来给出有针对性的建议或者显示广告。在这一点上，亚马逊已然做到了极致，他们为客户推荐的产品绝不是一个巧合。亚马逊的推荐引擎完全是基于客户在过去一段时间的购买行为：客户购买过的商品、客户的购物车中所收藏的商品、客户浏览过的商品、其他用户浏览或购买的商品。亚马逊通过分析和计算，为每位客户定制了专属的个人主页，帮助公司业务保持持续增长。在医疗保健领域，“谷歌流感趋势”项目依据网民搜索内容分析全球范围内流感等病疫传播状况，与美国疾病控制和预防中心提供的报告对比，追踪疾病的精确率达到97%。社交网络为许多慢性病患者提供临床症状交流和诊治经验分享的平台，医生借此可获得通常在医院得不到的临床效果统计数据。基于对人体基因的大数据分析，可以实现对症下药的个性化治疗。

可以看出，大数据这个概念发展到今天，已经不仅仅用来形容人类自身或者机器设备创造的大量非结构化和半结构化的海量数据，而更多的是指解决问题的一种方法，即通过收集、整理生活中方方面面的数据，并对其进行分析挖掘，进而从中获得有价值信息，最终衍化出一种新的商业模式。大数据可能带来的巨大价值正渐渐被人们认可，它通过技术的创新与发展，以及对数据的全面感知、收集、分析、共享，为人们提供了一种全新的看待世界的方法。更多地基于事实与数据做出决策，这样的思维方式，可以预见，将推动一些原本习惯于靠“差不多”或主观判断运行的社会领域发生巨大变革。

同样的，对于企业来讲，在当前这个信息空前融会贯通的时代，企业的市场竞争也变得复杂而深刻，技术将变成企业制胜的突破口。利用海量、多样的数据，实现快速、准确的采集和分析，将是企业实现业务模式创新、增强竞争力和提升绩效的有效方式。而能否借助 IT 手段让分散在企业不同系统里的数据流动起来，从而在实时变化的复杂市场环境中快速准确地做出决策，则是构成一家企业核心竞争力的重要因素。正因为如此，SAP HANA，这个“专注于实时大数据分析和应用的先进平台”，为企业提供了革命性的解决方案，它不仅能使企业实现大数据分析速度的百倍提升，及时获得更有价值的市场洞察；还能提供更全面的数据管理，为企业在激烈的市场竞争中抢得先机。不仅如此，借助 SAP HANA 的最新的支持包(截至本书出稿时，最新的更新包为 SPS06)，SAP 计划为企业数据中心部署 SAP HANA 提供更多支持，让数据中心实现全天候运行。此外，SAP 还将扩展与第三方备份工具的集成，旨在更好地与用户现有 IT 基础设施进行集成和整合，最大限度降低维护成

本。此外，SAP 还将增加安全增强功能，例如通过加密保护静态数据、增强型授权等功能加强系统安全性。

通过下面这个案例介绍，你将会对 SAP HANA 如何支持企业基于大数据的分析有更加深刻的理解。在 2012 年 SAP 全球技术研发者大会(拉斯维加斯)上，SAP 执行董事会成员、技术及平台产品负责人史维学博士介绍了最新的基于 PB 数量级的 SAP HANA 性能测试。该测试是在位于美国加利福尼亚州圣克拉拉市的英特尔数据托管中心进行的。在该中心 SAP 和 IBM 合作组建了由 100 台 IBM 服务器、100TB 内存以及 4000 个 CPU 内核组成的服务器集群①作为 SAP HANA 的硬件平台。测试的数据取样于多个 SAP NetWeaver BW 客户，并根据这些客户使用 BW 系统的实际情况，创建了由 1.2 万亿行、61 列的销售数据(即每天 330 万条交易数据，连续取 10 年)组成的大小为 1PB(1PB = 1024TB = 1 048 576GB)的原始二维资料表。该数据表能够真实地反映出目前这些取样客户的真实情况，因此测试结果非常具有代表性。由图 1-2 可以看出，SAP HANA 平台能够每小时支持 112 602 次查询操作，满足大于 5000 个查询及分析用户同时在线和并行查询的需求。并且由柱状图可以看出，大多数复杂的商业智能查询执行时间不超过 1 秒钟！而这样惊人的速度是在没有任何二次索引、物化视图②以及数据聚合的基础上取得的，即没有使用任何传统的数据查询加速技术。这种结果和性能在传统的、以硬盘存储为基础的数据库中是难以想象的。可以说，在这个测试平台上，只要数据被加载完毕，客户就几乎能在极短的时间内得到任何基于此 PB 级别数据表的查询结果。此外，他们不必提前数天要求数据库管理员或者开发人员构建结构来加快速度，也不需要等待数据库重建索引或者缓存。SAP HANA 在提供给他们提升了百倍的数据分析速度的同时，大大简化了操作流程。

图 1-2 是这次性能测试的一个概览结果。

---

① 服务器集群是指将很多服务器连接起来一起进行同一种服务，在客户端看来却像是只有一个服务器。集群可以利用多个计算机进行并行计算从而获得很高的计算速度，也可以用多个计算机做备份，从而在任何一个机器损坏的情况下还能保证整个系统正常运行。

② 物化视图，用于预先计算并保存表连接或聚集等耗时较多的操作的结果，这样，在执行查询时，就可以避免进行这些耗时的操作，从而快速地得到结果。物化视图有很多方面和索引很相似：使用物化视图的目的是为了提高查询性能；物化视图对应用透明，增加和删除物化视图不会影响应用程序中 SQL 语句的正确性和有效性；物化视图需要占用存储空间；当基表发生变化时，物化视图也应当刷新。

其中物化视图有三种：聚集物化视图、包含连接物化视图、嵌套物化视图。三种物化视图的快速刷新的限制条件有很大区别，而其他方面则区别不大。

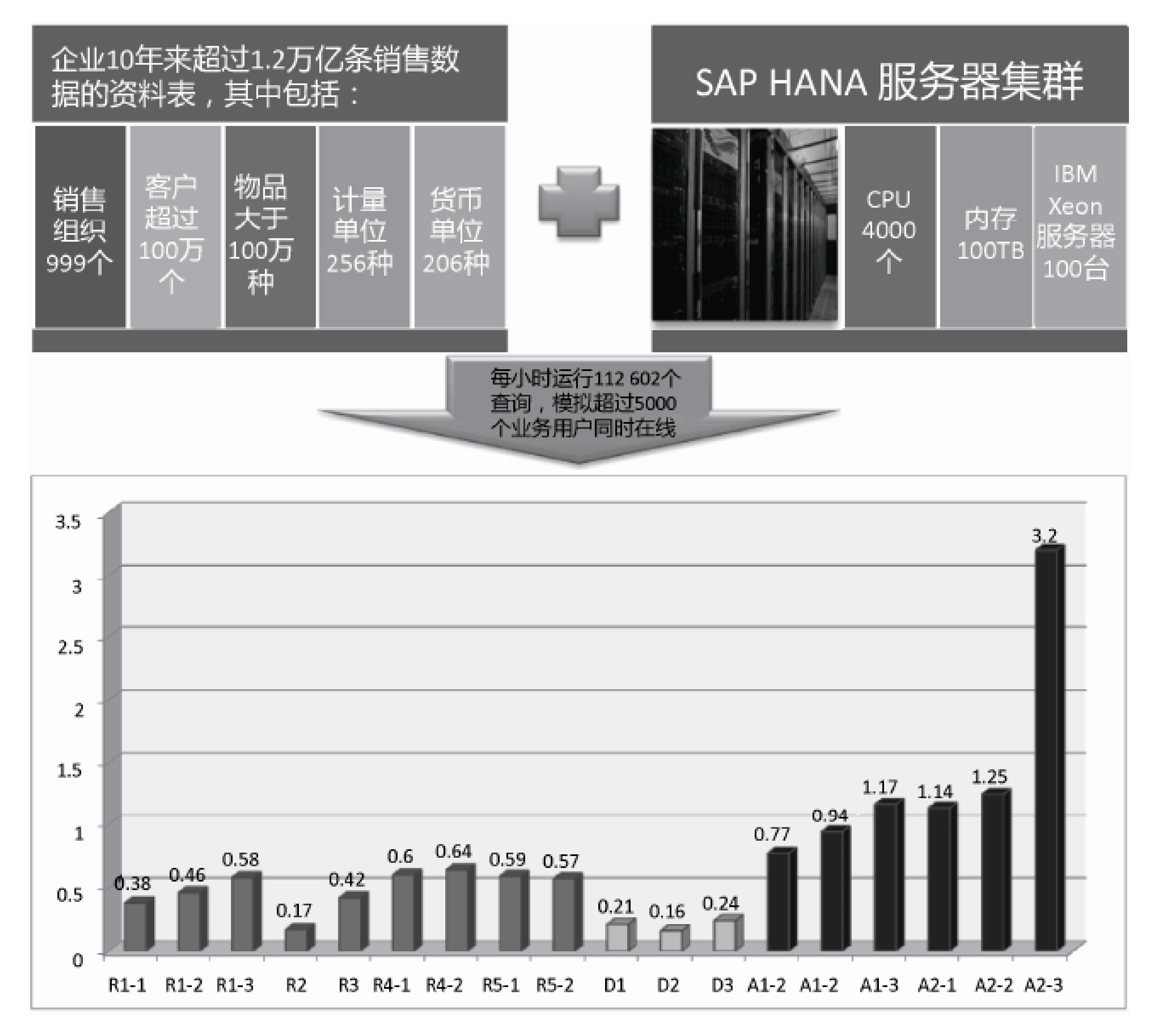


图 1-2

SAP HANA 平台在数据分析方面之所以能表现出如此令人吃惊的性能，得益于它强大的内存计算技术。而谈到 SAP 内存计算技术，则首先要了解一个数据库领域划时代的产品——SAP HANA 内存数据库。

第四节　SAP HANA 内存数据库

在讲 SAP HANA 内存数据库之前，我们先打个小比方。比如身在北京的你早上起来想喝杯牛奶，于是你买了一张飞往澳大利亚的机票，首先你先要从北京飞往香港等待几个小时来转机，然后才能飞往澳大利亚。终于你经过近 20 小时的飞行时间到了悉尼机场，接着你还要搭乘各种交通工具从机场辗转到牧场。可惜到达时已是半夜，你不得不再等几个小时到天亮后才能迎着晨曦喝到刚挤出来的新鲜的牛奶。看到这里你肯定会摇头说，姑且不计费用，这么做简直是谋杀我的时间！我直接去厨房的冰箱拿新鲜牛奶喝要不了 1 分钟的时间！

其实这个虚拟的故事主要是想说明，如果把 CPU 比做你，把数据比做新鲜牛奶，正如你在虚拟的故事里花费不可思议的时间才能喝到牛奶一样，CUP 在现实中的确要等待漫长的时间才能拿到所需的数据。因为服务器在处理数据请求时(例如一个查询操作)，CPU 首先会从其缓存(SRAM[①])中找数据，缓存中找不到，再从内存(DRAM)中找，内存里没有，再从硬盘(DISK)上找。在当下的各种存储产品中，按照数据传输速度从快到慢排序依次为内存＞闪存＞机械硬盘＞磁盘。磁盘目前主要用于数据备份，硬盘由于其低廉的价格(单位容量)以及良好的性能，一直以来是统治计算机系统存储的霸主，同时也是传统关系型数据库的主要存储设备。在企业级应用层面，分析员需要查询的绝大部分数据都储存在硬盘里面，CPU 需要经过层层中转(硬盘→硬盘缓存→内存→CPU 缓存→CPU)，才能得到需要的数据。因此，服务器系统性能的指标很大一部分取决于硬盘数据传输速度的快慢。尽管人们一直以来想了很多办法提高硬盘的 I/O[②]效率，如提高硬盘高速缓存的容量、将硬盘组成 RAID[③]、尽量保持硬盘顺序读写等，但是与 CPU 和内存的速度相比，硬盘的速度仍然是计算机系统性能最大的瓶颈。在图 1-3 中，我们比较了当前最快的基于 SAS 的服务器硬盘[④]

---

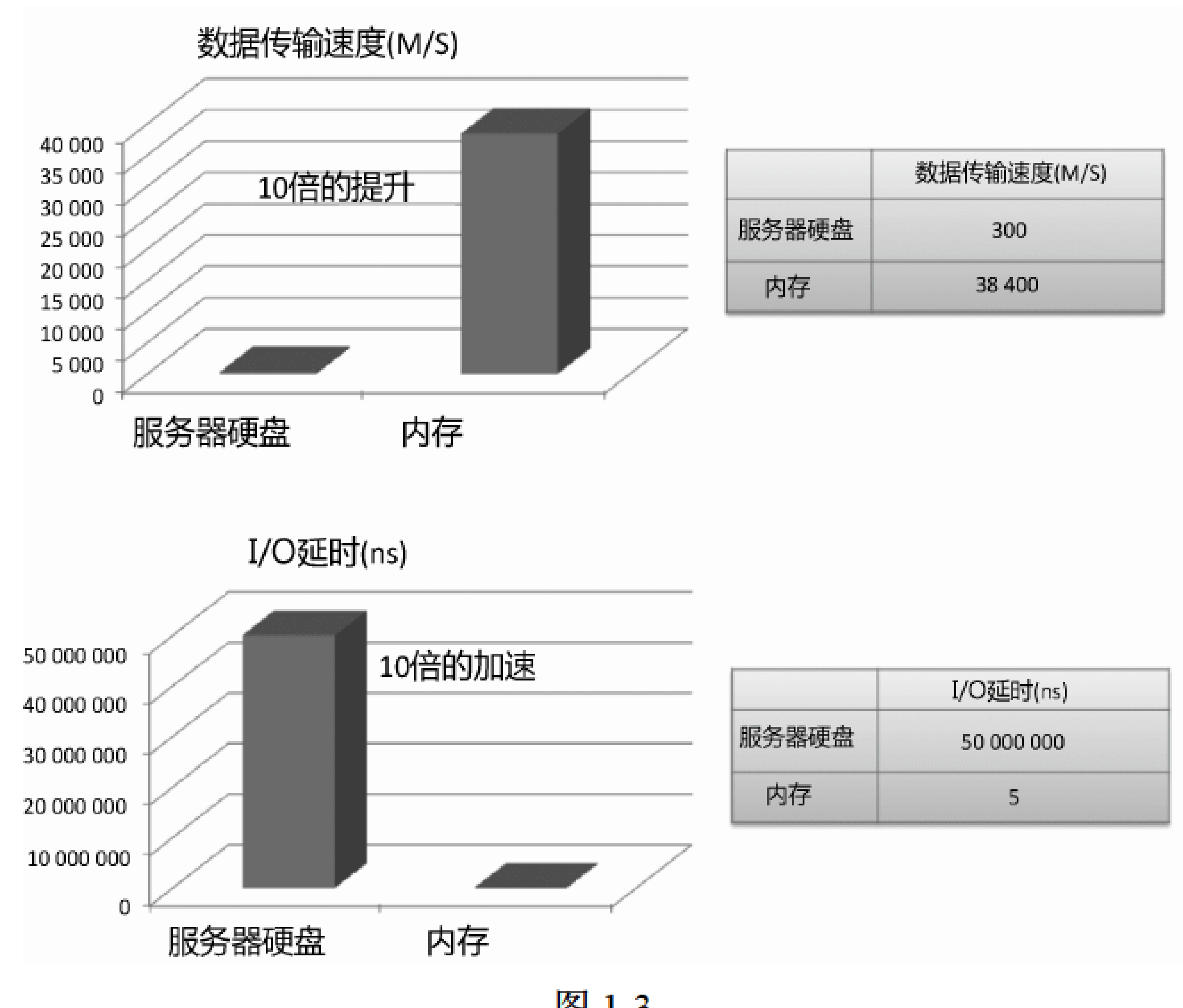
① ROM 和 RAM 指的都是半导体存储器，ROM 是 Read Only Memory 的缩写，RAM 是 Random Access Memory 的缩写。ROM 在系统停止供电的时候仍然可以保存数据，而 RAM 通常都是在断电之后就丢失数据，典型的 RAM 就是计算机的内存。
RAM 有两大类，一种称为静态 RAM(Static RAM/SRAM)，SRAM 速度非常快，是目前读写最快的存储设备。但是它也非常昂贵，所以只在要求很苛刻的地方使用，譬如 CPU 的一级缓冲，二级缓冲。另一种称为动态 RAM(Dynamic RAM/DRAM)，DRAM 保留数据的时间很短，速度也比 SRAM 慢，不过它还是比任何的 ROM 都快，但从价格上来说 DRAM 相比 SRAM 要便宜很多，计算机内存就是 DRAM 的。

② I/O，既输入/输出(Input/Output)，分为 I/O 设备和 I/O 接口两个部分。对磁盘的每个 I/O 就是在磁盘与一些 RAM 单元之间相互传送一些相邻扇区的内容。

③ RAID(Redundant Arrays of Inexpensive Disks)，即廉价冗余磁盘阵列。原理是将多台硬盘通过 RAID Controller(Hardware 或 Software)结合成虚拟单台大容量的硬盘使用。从 RAID 概念的提出到现在，已经发展了多个级别，明确标准级别分别是 0、1、2、3、4、5 等。但是最常用的是 0、1、3、5 四个级别。利用这项技术，可将数据切割成许多区段，分别存放在各个硬盘上。磁盘阵列还能利用同位检查，在数组中任一个硬盘故障时，仍可读出数据，在数据重构时，将数据经计算后重新置入新硬盘。

④ 目前，企业级高端服务器都使用 SAS 硬盘，该盘分为两种协议，即 SAS-1 及 SAS-2 接口，SAS-1 接口传输带宽为 3.0GB/s，转速有 7.2kr、10kr、15kr。现在 SAS-1 接口的服务器硬盘大部分已被 SAS-2 接口硬盘取代，该硬盘尺寸有 2.5 寸及 3.5 寸两种，接口传输带宽为 6.0GB/s，转速有 10k 转/分和 15k 转/分两种，常见容量为 300GB、600GB、900GB。SAS-2 的第二代更新，通常称为 SAS 2.1，改进了 SAS 设备的连接性能。

(SAS/15000 转)和主流内存(DDR3/2400)的速度差异。



	数据传输速度(M/S)
服务器硬盘	300
内存	38 400

	I/O延时(ns)
服务器硬盘	50 000 000
内存	5

图 1-3

是的，你没看错，内存与硬盘比较，传输速度提升了10倍(我们比较的是当前最先进的服务器硬盘，通常来讲，普通服务器硬盘的传输速度只有 80M/S，即内存传输速度要快于普通服务器硬盘 30 倍)，而平均 I/O 延时①加快了 100 万倍！因此，回到先前那个故事里面，CPU 从内存那里调取数据，就像你直接去厨房拿牛奶一样，而从硬盘那里调取数据，则不得不等待一段漫长的时间。根据 IDC 的一项调查报告的数据显示，全球服务器平均 80%的情况下处于空闲状态，这种无效的空转不仅影响 IT 系统的性能，也浪费了大量的能源。究其根本原因，就是因为硬盘的低传输速率和高 I/O 延时导致 CPU 大部分时间都在等待数据从硬盘传输过来。所以，在传统数据库查询中，硬盘设备始终是制约系统性能最大的瓶颈。那么让我们设想一下，如果我们把数据全部放在内存里面，而不用访问物理硬盘来做基于大量数据的查询和分析，将会产生怎样的效果呢？就像你直接从冰箱拿牛奶喝的愉快体验一样，对于

① I/O 延时(I/O Latency)，也称为 I/O 响应时间，是指从 CPU 内核对存储设备发出一个读或者写的 I/O 命令到 CPU 内核接收到回应的时间，I/O 响应时间包括 I/O 操作在存储设备中处理的时间和 I/O 操作在 I/O 等待队列中所花费的等待时间。

分析和决策者而言，那绝对是一种令人振奋的体验：以往需要几天甚至几周才能完成的报表，可能会在 1 秒钟就展现在他们的面前。

SAP HANA 内存数据库恰恰把我们的设想变成了现实。它是可以将企业的全部数据都放在服务器内存中直接进行操作的数据库。需要特别指出的是，我们这里指的“全部数据”，就是企业运营的所有数据，而不像其他一些数据仓库所做的，要先对原始数据进行筛选、聚合等处理后才存储进数据库。这样做的好处就是所有在 SAP HANA 内存数据库上进行的分析结果都是基于企业原始数据的，从而避免了以往数据仓库分析数据经过筛选和聚合后所带来的数据失真。因此，SAP HANA 内存数据库所带来的优越性能不仅仅在于 CPU 对内存读写比对硬盘读写快，更重要的是，SAP HANA 内存数据库从根本上抛弃了许多传统硬盘数据库管理的方式，以全部数据都放在内存中为基础，对数据管理进行了新的体系结构的设计，在数据缓存、快速算法、并行操作方面也进行了相应的改进，从而使数据处理速度比传统数据库的数据处理速度快很多，一般在 10 倍以上，理想情况甚至可以达到上千倍。举个例子，上海寰融信息技术有限公司(GFT)是一家 2010 年成立的金融软件平台公司。其产品的一大核心价值是为客户提供对金融类数据基于数学模型的决策分析，而且是穷尽式的分析。自从 GFT 公司向客户提供了一个基于 SAP HANA 平台技术的策略分析软件以后，GFT 客户的交易策略整个方式发生了革命性的变化。SAP HANA 强大的内存数据库和内存计算引擎，使得分析运算速度超快，其中最快的运算速度比客户过去的系统要快 1 万倍，过去需要几个小时得出的一个投资策略，现在几秒钟、几分钟就能完成。

对于 SAP HANA 内存数据库可以把企业的全部运行数据都放在内存中，或许你会有这样的疑问，目前企业级的数据库数据量往往以 TB 或者 PB 级计算，这么多数据能全部放到内存中吗？SAP 给出的答案是肯定的。因为把全部数据放到内存中的两个关键条件，即内存的价格问题和系统平台技术支持已经成熟。从内存价格角度来讲，十几年前的内存价格相对于硬盘来讲是相当昂贵的。以 2000 年为例，当时 1500 美元只能购买 1GB 内存。但是随着每 GB 内存价格随时间推移而不断下降(从图 1-4 中我们可以看到近 12 年来内存价格的变化曲线)，到了 2012 年，与 2000 年时相比，内存每 GB 的单价下降了近 240 倍。可以说以前非常昂贵的内存在今天已经可以被企业大规模采购并使用了。从系统平台的技术来讲，以往基于 32 位系统的 x86 服务器最大只能支持 4GB 的内存，而今天基于 64 位系统的 x86 服务器内存理论上可以达到几乎无限制的 18 EB(即 180 亿 GB)。目前最新的企业级服务器单机

最大已经能支持 2TB 的内存，而将这些服务器组成服务器集群后，企业服务器系统则能支持上百 TB 的内存。



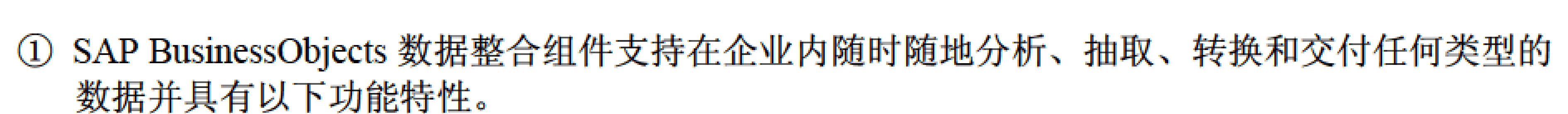
图 1-4

那么，SAP HANA 内存数据库都有哪些特点呢？下面我们来逐个分析一下：

一、不受数据类型限制

SAP HANA 内存数据库可以访问任何数据。当企业需要非 SAP 应用程序中的运营数据，或希望在现有分析模型的基础上进行扩展时，任何数据源均可作为 SAP HANA 的数据基础。通过 SAP HANA 平台集成的 SAP BusinessObjects 数据整合组件[①]，可以非常灵活地将非 SAP 运营数据快速加载到 SAP HANA 内存数据库中。这样，企业就可以通过极其精简的流程创建一个特定业务情景的完整视图。这一特性





① SAP BusinessObjects 数据整合组件支持在企业内随时随地分析、抽取、转换和交付任何类型的数据并具有以下功能特性。
- 完备的数据整合功能：访问和整合来自任何数据源的数据，并设计一个高效、可靠的数据整合流程。
- 本地文本数据处理：充分挖掘非结构化文本数据的含义，以此提高业务洞察力。
- 抽取、转换和加载(ETL)功能：实时地迁移和整合数据。
- 端到端的元数据管理：了解分散系统(从数据源到 BI 环境)中数据之间的影响和沿袭。
- 直观的界面：使用直观的拖放界面快速开发数据整合项目，其中的选项可导入数据质量功能。
- 企业级性能：通过并行、缓存和网格计算方法，迁移海量数据。

最大的好处在于，企业可以保留原有的分析应用，并将之快速部署到 SAP HANA 平台上而不用开发新的应用，为企业节省了大量的时间和费用。

二、不受数据量限制

早在 2012 年 SAP 中国商业同略会(2012 SAP SAPPHIRE China)期间，IBM 大中华区董事长及首席执行总裁钱大群先生表示，IBM 与 SAP 的合作已经有40 年历史，这一次 IBM 很荣幸地以100 个节点[100 台四路 Xeon E7(10 核心)服务器]、100TB 主内存、4000CPU 核心的新集群系统成为最强大的 SAP HANA 系统，证明了 SAP HANA 强大的扩充能力。与此同时，SAP 执行董事会成员、技术及平台产品负责人史维学博士介绍，该系统是当时世界上最大的内存计算数据库系统，也是世界上性能最快的内存计算系统。IBM 和 SAP 此次共同推出这套 SAP HANA 系统，其内存容量超过了大部分企业的数据产品的应用容量，得益于 SAP HANA 内存数据库优异的数据压缩技术(列存储，10 倍以上的数据压缩)，可以实现 PB 级别数据库的存储。

三、既可用做分析，也可以用做事务

SAP HANA 内存数据库是一个内存数据库的混血儿，在技术层面，它同时支持 OLTP 和 OLAP 技术。在存储层面，它不仅包含行存储，也包含列存储，而且还支持基于对象存储的数据库技术。但仅从这些话中，你可能无法体会到其开创性的变革之处。而看完下面的解释之后，你可能会由衷地说：“Wow，SAP HANA 内存数据库真的非常了不起！”

首先我们要先了解两个名词，它们分别是“OLTP”和“OLAP”。

OLTP(On-Line Transaction Processing)，联机事务处理。OLTP 是传统的关系型数据库的主要应用。其主要用于基本的、日常的事务处理，涵盖了一个组织的大部分日常操作，如购买、库存、制造、银行、工资、注册、记账等，它是面向顾客的，用于办事员、客户和信息技术专业人员的事务和查询处理操作。在过去几十年的时间里，OLTP 技术已经发展得比较成熟，其基本特征是用户的原始数据可以立即传送到计算中心进行处理，并在很短的时间内给出处理结果。这样做的最大优点是计算机系统可以即时处理输入的数据，并及时地做出反应。衡量联机事务处理系统的一个重要指标是系统性能，具体体现为实时响应时间(Response Time)，即用户在终端上输入数据之后，到计算机对这个请求给出答复所需要的时间。OLTP 是由



数据库引擎负责完成的，旨在使事务应用程序仅写入所需的数据，以便尽快处理单个事务。支持 OLTP 的数据库通常具有以下特征：

- 支持大量并发用户定期添加和修改数据；
- 反映随时变化的单位状态，但不保存其历史记录；
- 包含大量数据，其中包括用于验证事务的大量数据；
- 具有复杂的结构；
- 可以进行优化以对事务活动做出响应；
- 提供用于支持单位日常运营的技术基础结构；
- 个别事务能够很快地完成，并且只需访问相对较少的数据；
- OLTP 系统旨在处理同时输入的成百上千条事务。

由这些特性可以看出，由于 OLTP 技术更偏向于处理大量并发的对数据库记录的添加、修改、删除等访问量较少、需要很快完成的事务，故而如果用于查询操作，其只能满足一些简单的、数据量较小的查询。例如你在 ATM 柜员机查询你的账户余额，事务员查询某笔订单的状态等。如果你要做银行整体的现金流报表或者看企业整体的销售与库存关系的查询分析，那么 OLTP 是不适用的。如果我们要对企业的大量业务数据进行复杂的查询分析怎么办？这就要用到下面要说的 OLAP 技术了。

OLAP(On-Line Analytical Processing)，联机分析处理。在过去的二十多年中，大量的企业利用关系型数据库来存储和管理业务数据，并建立相应的应用系统来支持日常业务运作。这种应用以支持业务处理为主要目的，即我们前面提到的 OLTP，它所存储的数据被称为操作数据或者业务数据。随着市场竞争的日趋激烈，企业更加强调决策的及时性和准确性，这使得以支持决策管理分析为主要目的的应用迅速崛起，这类应用主要用来弥补关系数据库管理系统支持的不足，统一分散的公共应用逻辑，并在短时间内响应非数据处理专业人员的复杂查询要求。与 OLTP 技术面向的主要是事务操作人员不同，OLAP 技术是面向分析人员、管理人员的，其目的旨在提供分析人员、管理人员快速直观访问数据的一种途径，使分析人员、管理人员能直观地从大量数据中获得有用信息以提供决策依据。通常来讲，OLAP 技术对数据库访问也有别于 OLTP 不断地对数据库进行增删改操作。它是不涉及数据库操作的，即访问是只读的，但是这种访问不是简单地记录属性的检索，而是为了从数据中获取有用信息而针对大量数据的查询，往往一次需要查询上百万条以上数据。正因为如此，联机分析处理具有灵活的分析功能、直观的数据操作和分析结果可视化等突出优点，从而使用户对基于大量复杂数据的分析变得轻松而高



效，以利于迅速做出正确判断。它可用于证实人们提出的复杂的假设，其结果是以图形或者表格的形式来表示对信息的总结。

通过前面对 OLTP 和 OLAP 的讲解，我们可以了解到这两种数据处理技术在数据来源、数据内容、数据模式、服务对象、访问方式、事务管理等方面都有不同的特点和要求。由于传统的基于 OLTP 技术的关系型数据库偏重于企业日常事务处理工作，通常是对单一或一组记录进行修改和查询，它注重的是如何在多并发操作时减少响应时间，保证数据的安全性和完整性。而对于 OLAP 技术来讲，由于制定决策分析要对大量的历史数据进行复杂查询和分析，因此分析性处理操作可能要连续运行几个小时甚至几天的时间来完成这些工作，从而消耗了大量的系统资源，对系统性能方面有非常大的影响。与此同时，为了使分析结果更加准确，决策分析型处理需要大量的基础数据，这些数据有来自企业内部的，也有来自企业外部的。来自企业外部的数据又可能来自不同的数据库系统，在分析时如果直接对这些数据进行操作会造成分析的混乱。对于外部数据中的一些非结构化数据，传统的关系型数据库常常无能为力。

为了解决这个问题，20 世纪 80 年代中期诞生了数据仓库的概念。到了 20 世纪 90 年代，数据仓库已从早期的探索阶段走向实用阶段。业界公认的数据仓库概念创始人比尔·恩门(W. H. Inmon)在《数据仓库》(*Building the Data Warehouse*)一书中对数据仓库的定义是“数据仓库是支持管理决策过程的、面向主题的、集成的、随时间变化的、持久的数据集合”。构建数据仓库的过程就是根据预先设计好的逻辑模式，从分布在企业内部各处的 OLTP 数据库中提取数据，并对其经过必要的变换最终形成全企业统一模式数据的过程。当前，数据仓库的核心仍是 RDBMS[①]管理下的一个数据库系统，通过这个数据库系统，OLAP 服务器可以为一些前端工具或应用(比如报表或 BI 系统)提供服务。从图 1-5 可以看出，企业为了满足事务和分析处理的不同要求，往往要建立两套不同的数据库系统，即为企业日常运作提供支持的基于 OLTP 的关系型数据库系统和为企业制定决策分析准备数据的数据仓库系统。而在流程图的最后，作为直接面向决策者的前端工具需要非常复杂的步骤(业务数据→提取、转换、上载→数据仓库→OLAP 服务器→前端应用)才能得到最基础的业务数据，以至于对于新录入的数据，有时甚至要等待几天或者几周才能被反映到前端应

① RDBMS(Relational Database Management System)，关系型数据库管理系统。

用，所以基于这种模式的分析和决策无法达到真正的实时。因此无论从成本角度还是时间角度，其对于企业来说需要的付出都是非常巨大的。

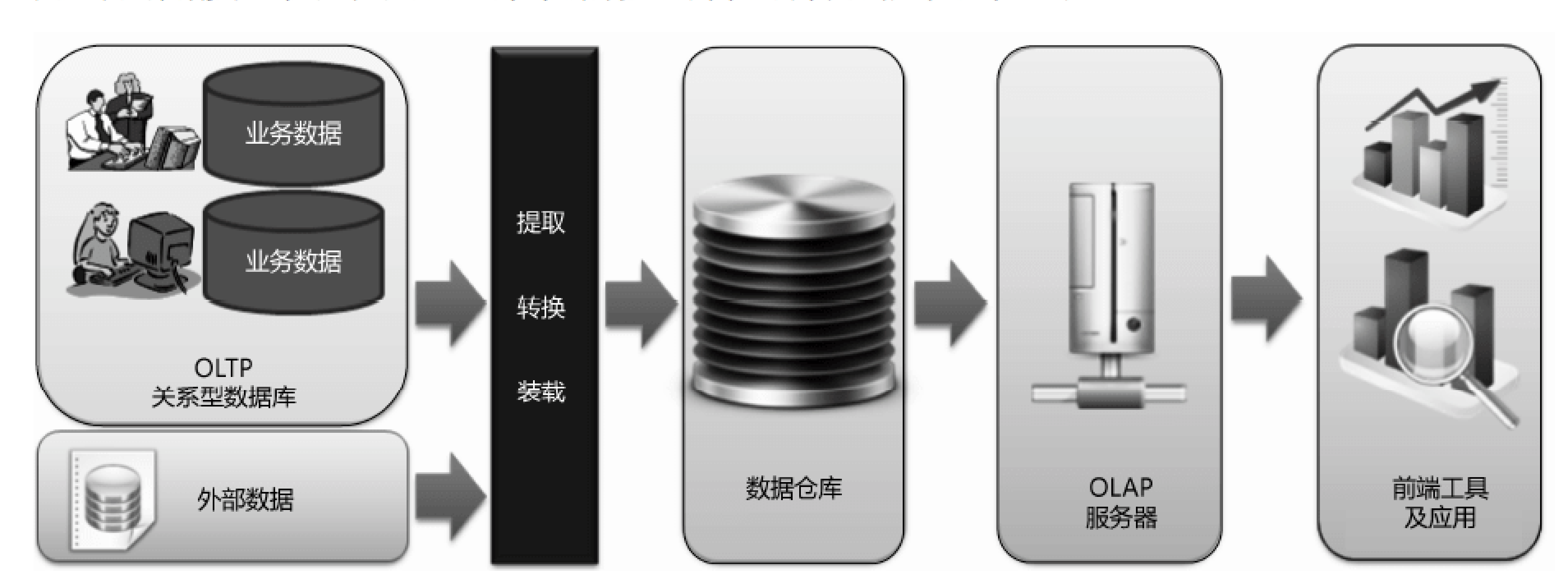


图 1-5

针对这个问题，SAP HANA 平台开创性地把用于事务型处理(OLTP)和用于分析型处理(OLAP)的数据全部放在同一个内存数据库中。这一变革彻底消除了以往企业日常业务运行数据和决策分析数据需要分别处理的壁垒，大大提高了企业事务处理的速度，简化了前端工具取得数据的步骤和时间，使用户通过基于 SAP HANA 平台的前端工具能真正做到实时分析而无需等待。而这一特性，也是 SAP HANA 所具有的变革性的特性之一。与此同时，正是由于 SAP HANA 内存数据库能很好胜任传统数据库的存储和访问工作，这种传统和创新型技术的结合使得开发人员可以为程序选择最好的技术，并且在需要时二者可以并用。



综上所述，SAP HANA 内存数据库从概念上来说是通过利用内存数据存储提升数据库查询执行速度的软硬件结合体，与此同时，SAP HANA 内存数据库也能很好地胜任传统数据库的存储和访问，例如它也有基于行式存储的表可供使用。因其不受数据量、数据种类的限制，既可用做分析，也可以用做事务；没有数据缓存，也不需要任何调校，或者是预先聚合、预先处理的特性，意味着开发人员可以基于 SAP HANA 平台访问任何数据源，开发基于任何存储方式的应用。它不需要像传统数据库那样用复杂的编程技巧，如提前计算值(物化聚集)或者创建数据构架来加速基于大量数据的查询访问。这种全新的数据库体验，带给数据库应用开发的灵活性和简易性将是无可比拟的。并且，SAP HANA 内存数据库为了保持用户操作尽可能小地对数据库产生改变，采用了只对原始数据库的增量变化进行记录的方式。其数据是增加或插入到一个表列而不是就地修改，这种方式带来的好处不只是速度上的

提升，而且由于保留了所有的旧数据，你的应用程序可以随时高效地访问数据记录在所有时间段内的数值，并可以依此提供该记录随时间变化的数据视图。

第五节　SAP 内存计算

在讲 SAP 内存计算之前，我们先花点时间看看图 1-6，这张图表示传统的数据库应用，例如 SAP 的 ERP 解决方案，和数据库之间的交互与基于 SAP HANA 平台的应用和数据库交互方式上的不同。

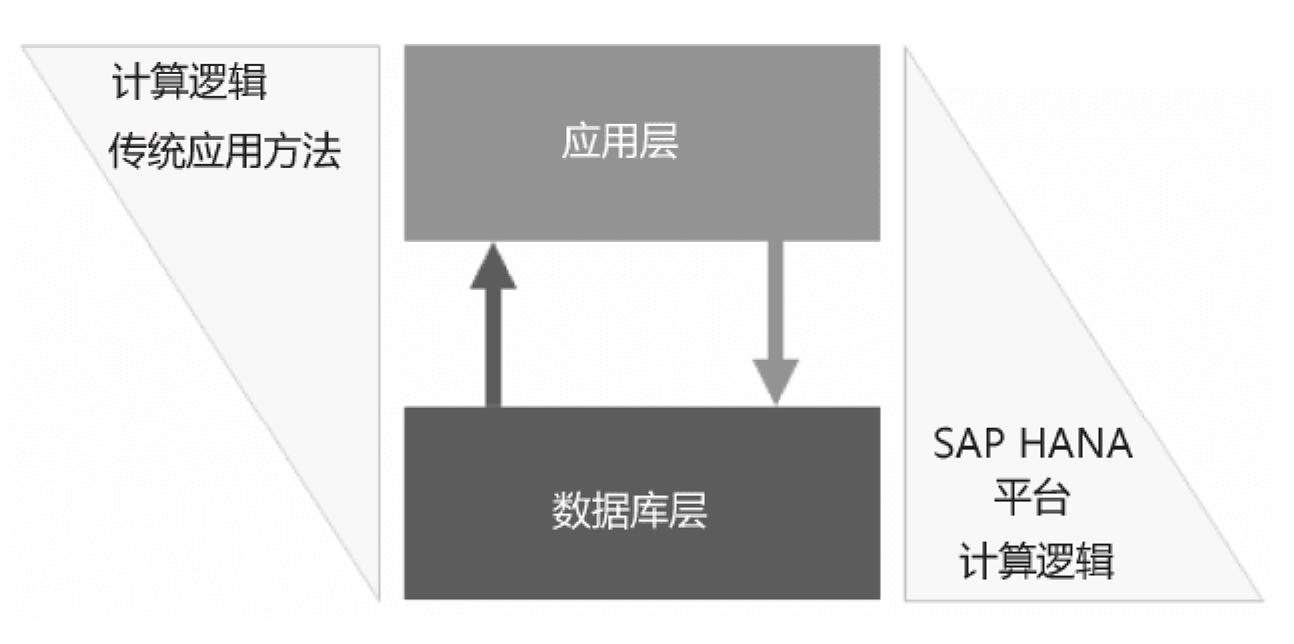


图 1-6

从这张图可以看出，不管是哪种交互，数据都在应用层和数据库层两个层之间进行交换。不同的是，在传统的应用与数据库交互方式中，应用层承担了大部分的计算工作，而数据库的作用主要是数据的载体，只在应用层需要数据时才把所需的全部数据传送过去，在数据库层中不会有任何的运算发生。这样做非常大的弊端在于，每次应用层提出数据请求，数据库就不得不把大量原始数据全部传输过去进行计算，而且越是复杂的查询，传输的数据量就越大，数据量越大，所需花费的传输时间越多，导致应用层的计算不得不等待所有数据加载完毕后才能进行。因此，我们常常会有这样的体验，点开一个数据量很大的报表或者执行一次很复杂的查询操作，往往要等待半天才能得到想要的结果。而在 SAP HANA 平台上，这种情况是不会出现的，功臣就是 SAP 内存计算技术。SAP HANA 内存计算创新性地把原本是应用层应做的数据运算下沉到数据库层面来做，而应用层只需等待数据库层面返回运行结果并把它展示出来即可。这样一来，传输到应用层的数据大大减少，而体现



出来的结果就是终端用户可以快速得到其所要的运算结果，大大提升了用户体验。

因此，SAP 内存计算指的就是基于服务器内存，快速处理大量原始的、非聚合的数据，并为查询或事务提供实时结果的一种技术。我们前面所提到的关于 SAP HANA 的所有案例里面，不约而同地都提到了一个字，那就是“快”。“快”是 SAP HANA 平台给人的最直接的感受，也是 SAP HANA 平台宣传的重点，而这个“快”字，从根本上取决于“SAP 内存计算”这个 SAP HANA 平台的核心技术。那么 SAP 内存计算是怎么做到的呢？它又包含哪些方面呢？接下来，让我们详细了解一下 SAP 内存计算技术吧。

一、“快”之先决条件——把数据放在 CPU 感觉最“顺手”的位置

如果你参观过饭店的厨房，你会看到厨师在拿到点菜单前其实已经把原料都在厨房准备好并按顺序排放整齐了，这样做是为了使烹饪的原料都在自己最顺手的位置以保证出菜速度快，从而提高顾客的满意度。你能想象在你点完菜后，厨师要先跑到超市进行采购或者手忙脚乱找调料这种不可思议的事情吗？

正如厨师准备原料一样，如果我们想让 CPU 以最快的性能处理数据，我们首先要把数据准备成它最容易操作的模式，即“近”——触手可得，“易”——有条不紊。对于“近”，在我们介绍 SAP HANA 内存数据库时已经讲过，内存的读写 I/O 延时比硬盘快了 100 万倍，SAP HANA 内存数据库可以把企业运营的全部数据都放在内存里面，因此“近”我们已经实现了。那“易”呢？对于“易”来讲，我们就要考虑 CPU 缓存和内存之间如何优化了。

CPU 缓存(Cache Memory)是位于 CPU 与内存之间的临时存储器，它的容量比内存小得多但是与 CPU 交换数据的速度却比内存要快得多。因为 CPU 运算速度要比内存读写速度快很多，这样会使 CPU 花费很长时间等待数据到来或把数据写入内存，缓存的出现主要就是为了解决 CPU 运算速度与内存读写速度不匹配的这一矛盾。如图 1-7 所示，在应用提出数据处理需求时，CPU 会优先从缓存中搜索所需的数据，因此会出现两种情况，即缓存命中——CPU 找到所需数据，以及缓存未命中——CPU 搜索了缓存，但没有发现数据，因此数据必须从缓存外的存储设备读取。由于搜索缓存需要花费时间，所以缓存未命中增加了 I/O 操作的时间。如果缓存实现不理想，将产生很高的未命中率，由于每次缓存未命中都需要花费额外的时间，所以其将导致系统性能的下降。因此，SAP 内存计算技术所做的，首先就是让

CPU 缓存知道存储在内存中的数据结构是什么，分布在哪些位置，以及根据程序的运行，哪些数据短时间内会被 CPU 访问并提前将其加载过来。同时当 CPU 调用大量数据时，其可最大限度地保证 CPU 所需要处理的全部数据都能在缓存中找到，避免 CPU 去服务器整个内存中对大量数据寻址而浪费过多的时间，从而加快处理速度。



图 1-7

二、“快”之基础——列存储

SAP HANA 的内存数据库是一个通用的数据库平台，它能够在一个数据库中存储行、列以及文本模式的数据。这种通用平台的好处是开发者可以根据自身的需要来在行存储和列存储两种模式下自由切换开发模式。但是我们认为，对于常用的分析和查询应用来说，我们不需要读取数据表中的全部数据，可能仅仅一部分数据就能满足查询的需要。例如我们想按季度分析或查询销售总额，只需要销售订单时间和销售金额这两列数据即可，而销售数据表的其他列数据不会参与到查询操作中来，即这些部分可以忽略。在传统的行存储模式中，为了完成这一操作，必须要把数据表中的数据全部读取到缓存中，并由 CPU 来进行数据处理，即便是对结果不重要的数据，也必须通过 CPU 来判断是否需要参加运算，这就大大增加了 CPU 的负荷。而对于列存储模型，由于其给定列的元素是存储在一起的，CPU 可以快速地对指定列进行聚合类的操作，因此企业级应用常见的聚集操作在列存储模型中的速度要比在行存储模型中快得多。



与此同时，SAP HANA 内存数据库的列存储还能提供高效的数据压缩技术。特别是在数据库某一列具有如下特性：

- 具有相同的数据类型；
- 大多数数据值为很少的唯一值，例如国家、性别等。

数据压缩技术在 SAP HANA 内存数据库中的使用，一方面节省了数据存储的空间，使得单位内存能存储更多的数据，即提高了其信息密度；另一方面，由于信息密度的提高，CPU 在请求数据时能在更短的时间内得到更多的数据，从而提高了其性能。

到这里，如果你以为高查询性能以及高压缩率就是列存储的全部优点，那请你继续耐心看下去。因为随着现代计算机硬件技术的发展，支持多核心和多线程的 CPU 不断涌现，列存储还有一个非常大的优点体现出来，那就是基于列存储的“并行计算”。

三、“快”之利器——并行计算处理

在 2005 年以前我们所说的处理器性能的提高，主要靠提高主频来实现，即 CPU 在每个时钟周期内可以执行更多的指令，但缺点是功耗也随之上升，并且功耗提升是主频的立方关系。打个比方，英特尔 3.4GHz 的奔腾 4 至尊版，晶体管达 1.78 亿个，最高功耗已达 135 瓦。实际上，在奔腾 4 推出后不久，就在批评家那里获得了“电炉”的称号，更有好事者用它来玩煎蛋的游戏。很显然，当晶体管数量增加导致功耗增长超过性能增长速度后，处理器的可靠性就会受到致命性的影响。因此，英特尔开发了超线程技术，允许一个处理内核可以在同一时间执行两条指令线程，从而更好地利用 CPU 的片上资源。此后，从 2005 年开始，CPU 厂商开始了多核处理器的设计，即每个处理器内包含多个相互独立的计算内核。如今，基于多核心和超线程技术的处理器已经相当成熟，并被服务器厂商广泛应用。



综上所述，由于单个 CPU 的核心速度不会在短时间内被更快地提高，但是 CPU 的核心数却有望每 18 个月翻一番，因此如何利用多核 CPU 的并行处理功能已经对于未来的软件开发提出了挑战，甚至成为至关重要的命题。在 SAP HANA 内存计算管理中，其内存列存储不仅可以将给定列中的数据一起存储在内存中，从而便于分配一个或者多个 CPU 核心来处理单列，即垂直分片；也可以将数据表数据行进行集合，然后再分配至不同的处理器，即水平分片。对于单个多核心服务器或者多

台服务器集群，SAP HANA 内存计算管理都可以实现上述的两种方法。

在这里，我们推荐大家阅读由 SAP 公司创始人哈索博士编写的《内存数据管理》一书，你会对本小节的内容有更加深入的了解。

第六节 SAP HANA 能做什么

前面讲了很多关于 SAP HANA 的基本概念，现在我们看看 SAP HANA 到底能做什么以及能给企业带来哪些方面的收益。

- SAP HANA 提供多用途的内存应用设备，企业可以利用它即时掌握业务运营情况，从而对所有可用的数据进行分析，并对快速变化的业务环境做出迅速响应。使用 SAP HANA，企业可以即时访问相关信息，更快做出更加可靠的决策，并降低获取洞察力时对 IT 部门的依赖。
- SAP HANA 提供灵活、节约、高效、实时的方法管理海量数据。利用 SAP HANA，企业可以不必运行多个数据仓库、运营和分析系统，从而削减相关的硬件和维护成本。SAP HANA 将在内存技术基础上，为新的创新应用程序奠定技术基础，支持更高效的业务应用程序，如计划、预测、运营绩效和模拟解决方案。
- SAP HANA 可直接访问运营数据，而不影响 SAP ERP 和其他运营系统的性能。企业可以近乎实时地将主要交易表同步到内存中，以便在分析或查找时能够对这些表进行轻松访问。一旦数据可通过内存访问，各个部门就可以从预订单据、销售线索、服务要求等大量列表中查找单个行项目，而不会对运营系统造成任何影响。这种高效的建模流程支持提供明细行项目的直接访问模型，也支持更为复杂的分析流程的分析模型。
- SAP HANA 提供从概念到分析的高效工作流程。其可涵盖整个流程，从识别相关运营数据(将原始数据转化为相关信息)开始，到在模型中生成按语义分组的信息，最后是发布完成的模型。SAP HANA 与传统分析模型的主要区别在于其摒弃了任何物质化的东西，即所有模型都是完全虚拟的，均基于基本的具体运营数据计算结果。这样，模型就能够被方便修改。



- SAP HANA 可以访问任何数据。当企业需要非 SAP 应用程序中的运营数据，或想在现有分析模型的基础上进行扩展时，任何数据源均可作为 SAP HANA 的数据基础。使用 SAP BusinessObjects 数据服务组件，可以将非 SAP 运营数据加载到 SAP HANA 内存中，这样，企业就可以通过极其精简的流程创建一个特定业务情景的完整视图。
- SAP HANA 添加了易用的建模经验来进一步提高业务用户的自主性。视图遵循语义规则，将原始运营数据转化成可以理解的信息，据此，业务用户可以在基于 Web 的建模环境中自主地创建新分析模型。通过 SAP HANA，企业可以在业务运作期间基于海量的实时详细信息分析业务运营情况。企业可以探索和分析来源于所有数据源的全部交易数据和分析数据。运营数据在产生时由内存获取，并通过灵活的视图迅速将分析信息呈现给用户。外部数据可轻松地被添加至分析模型，与整个企业的数据进行整合。
- SAP BusinessObjects 的商务智能分析工具可以直接使用 SAP HANA 内存数据，使业务用户能够全面利用其所有高性能应用程序的洞察和分析功能。但是，如果用户希望使用 Excel 或其他工具和应用程序进行数据分析，那么他们可以通过 MDX、SQL 等标准接口连接到 SAP HANA。
- SAP HANA 为现有应用程序、运营系统或其他业务应用程序提供标准接口。这意味着 SAP HANA 不会因为连接到现有数据源而打乱现有系统架构，并且可以轻松利用现有 BI 客户端。作为一款完备的实时分析解决方案，SAP HANA 可以帮助企业尽快获得收益。



第二章　SAP HANA 基础进阶

在前面章节，我们了解了在大数据背景下 SAP HANA 内存计算平台的诞生和 SAP HANA 的一些基本概念。这一章我们会对 SAP HANA 平台做进一步详细讲解，然后介绍一些 SAP HANA 相关的知识。这样大家在听到诸如 SAP HANA 报表加速、SAP Business Suite on HANA、SAP BW on HANA、基于 SAP HANA 的分析应用、原生应用开发、Side-by-Side 方式以及 SAP HANA 企业云这些词汇的时候，能够有一个大概的背景了解。

第一节　SAP HANA 平台详解

大家如果打开 SAP HANA 的官方网站 www.saphana.com，可以看到如下描述 SAP HANA 的语句：

- “SAP HANA 是一个完全重新为实时业务构想的平台。”(SAP HANA is a completely re-imagined platform for real-time business.)
- “SAP HANA 将数据库和应用程序平台整合在一起并全部下放到内存中来实现对交易数据转换、分析、文本分析、预测等业务的实时操作。”(SAP HANA converges database and application platform capabilities in-memory to transform transactions, analytics, text analysis, predictive and spatial processing so businesses can operate in real-time.)

从上述描述中，我们看到 SAP HANA 的产品定位就是实时性分析平台。图 2-1 很好地描述了 SAP HANA 平台的组成，下面我们将对其各个部分做详细解释。

首先我们看看图 2-1 的左边部分，它列出了基于 SAP HANA 的应用可以扩展部署的各类终端，在这里我们不仅可以看到 SAP HANA 对于传统计算机的支持，还能看到其对于移动终端设备也能很好支持。

在图 2-1 的底部，我们能看到 SAP HANA 支持的用于数据集成的数据源。从图



中可以看出，SAP HANA 平台可以集成整合不同类型的数据源，包括结构化的交易数据(Transaction)、非结构化的数据(Unstructured)、地理位置信息(Locations)、物理机器上的数据(Machine)等，也可和流行的大数据框架比如 Hadoop 集成。

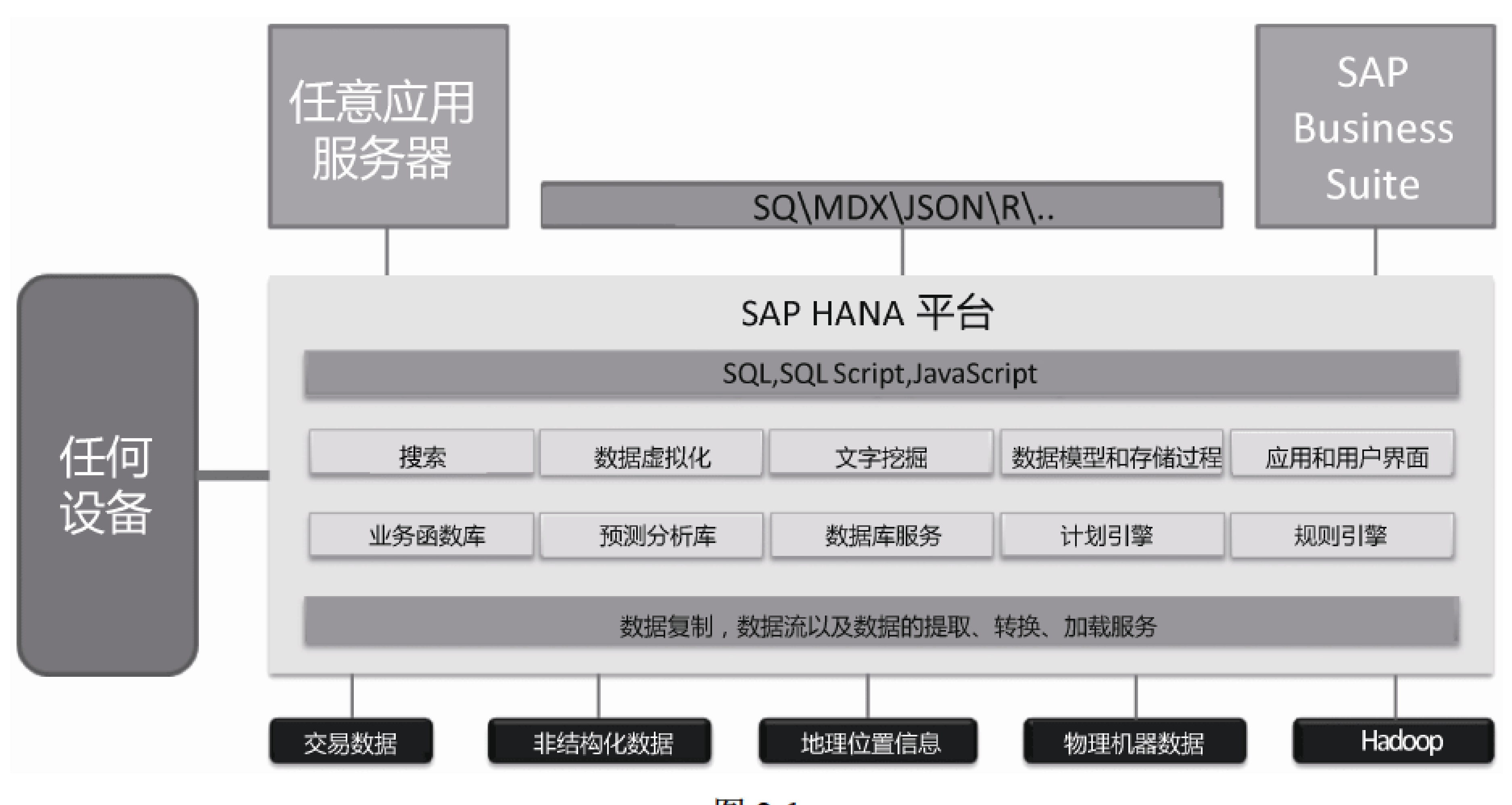



图 2-1

在图 2-1 的中部，我们能看到在将不同的数据源通过 ETL 或者其他工具导入 SAP HANA 平台后，SAP HANA 可以通过各种组件去实现不同数据处理、分析、计算和预测。

- 搜索：可以支持文本的模糊搜索(fuzzy search)，类似常用的搜索引擎。
- 数据虚拟化：可以直接在 SAP HANA 中建其他异构数据库的表(比如我们可以在 SAP HANA Studio 中创建基于 Sybase IQ 数据源的表或者 Hadoop 数据源的表)就像 SAP HANA 自身的表一样使用。
- 文本挖掘：可对文本进行挖掘和分析，比如把文本按话题进行分类，找出文本中的关键热点，或做文本的情感分析(比如可以对 weibo 数据进行情感分析，看内容是正面评价，还是负面，或是中立)。
- 数据模型和存储过程：从 SAP HANA 诞生就一直是主要部分的数据模型和存储过程，用于大数据建模分析和计算逻辑的实现。



- 应用和用户界面：可以帮助开发人员更快地创建应用的框架和服务，比如内嵌了轻量级的应用服务器(XS engine)、可以直接使用的 Web UI 控件(Site，navigation 支持，Widget 和 CSS 样式等)。
- 业务函数库：是一组可以重用的用 C++语言编写的函数库，主要用于财务方面。
- 预测分析库：和 BFL 类似，也是内嵌可以重用的函数库，主要用于分析和预测，比如 K-Means 算法、Association 分析、Multiple Liner Regression 算法等。
- 数据库服务：SAP HANA 用做基本数据库服务，比如日志和持久层服务等。
- 计划引擎：主要是用于支持应用服务器(planning application，比如财务计划应用或者按需计划应用)去执行基本的计划命令操作，比如 COPY/SET DATA/DELETE/DISGGREGATE 等命令操作。
- 规则引擎：支持自定义规则，并根据规则在 SAP HANA 生成 SQL Script 代码。
- SQL、SQL Script、JavaScript 是 SAP HANA 平台实现数据库层面的业务逻辑和大数据计算的主要计算语言。

在图 2-1 的上部我们能看到，SAP HANA 通过上面各个引擎处理完数据后，可以通过很多的外部接口将数据提供给应用层，比如通过 SQL、MDX、R、JSON 等。SAP HANA 不仅可以和 SAP Business Suite 的应用服务器集成，也可以通过 JDBC/ODBC 和其他的应用服务器集成。

第二节　SAP HANA 组件架构

在前面介绍完 SAP HANA 平台后，接下来我们在这一小节中介绍一下 SAP HANA 的主要组件和架构。

如图 2-2 所示，SAP HANA 系统主要包含的组件是 Index Server、XS Server、Name Server、Statistics Server、Preprocessor Server 等。

- 索引伺服器：SAP HANA 最主要的组件，支持实际的数据存储、数据处理和计算。



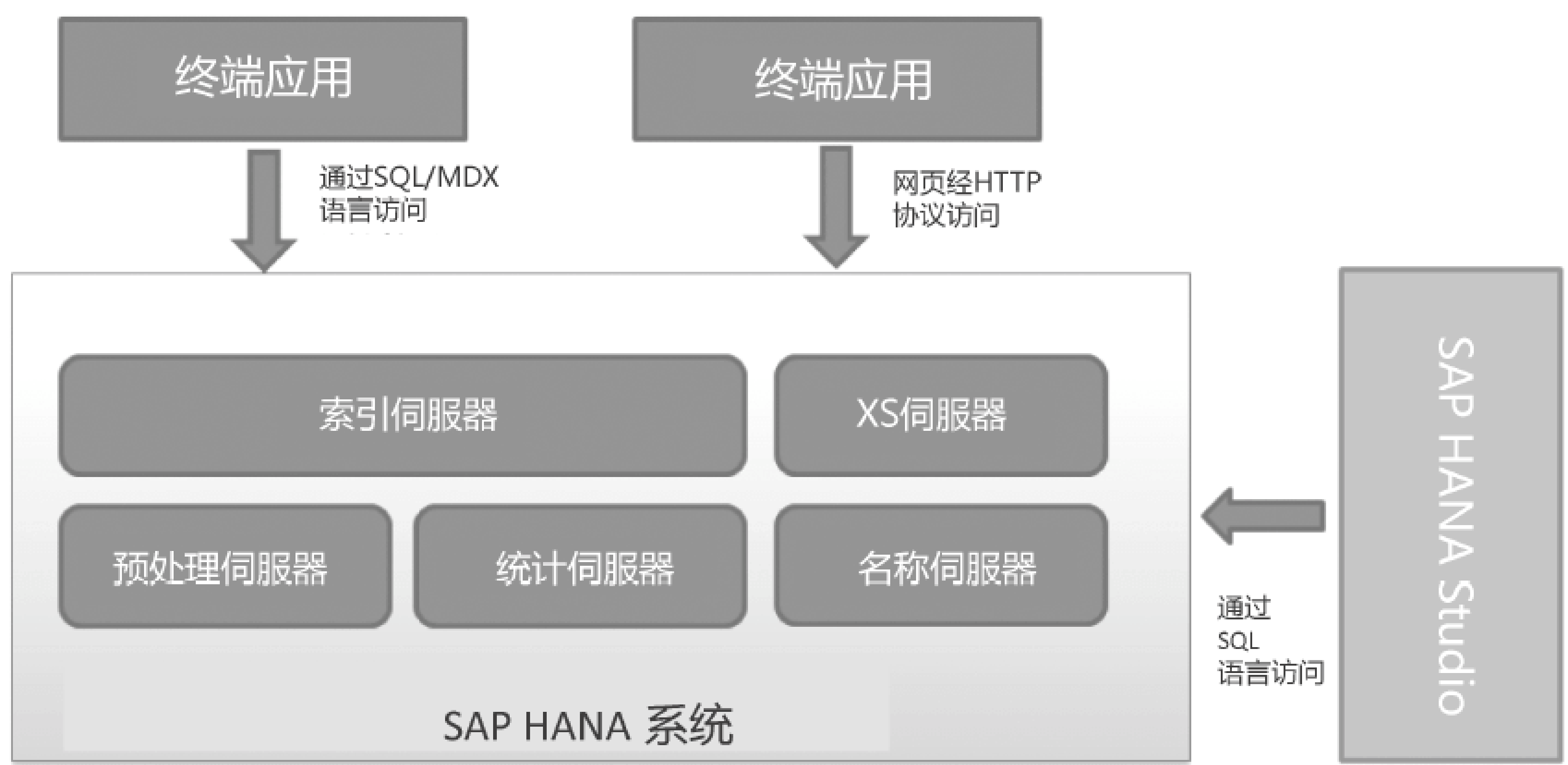


图 2-2

- 预处理伺服器：用来预处理非机构化数据，比如文本数据，并提取关键信息，这些信息后面也会被非结构化的组件用到，比如 Text Search。
- 名称伺服器：主要是包含 SAP HANA 系统的拓扑信息，在有多个 SAP HANA 系统(比如集群)的情况下，SAP HANA 通过名称伺服器知道哪些数据运用在哪个索引伺服器上。
- XS 伺服器：是 SAP HANA 系统从 SPS 05 开始的一个扩展，是一个轻量级的应用服务器，和索引伺服器紧密集成，通过 HTTP 协议客户端(Client)就可以访问 SAP HANA 的数据。
- 统计伺服器：主要收集 SAP HANA 的性能、状态、资源消耗方面的数据。通过 Studio 我们可以查看监控 SAP HANA 的系统运行状况。

另外常提到的有 SAP HANA Studio 和 SAP HANA Client。

- SAP HANA Studio：基于 Eclipse 的 HANA 开发和系统管理工具。SAP HANA 系统管理，HANA 建模，存储过程开发都是在 SAP HANA Studio 上进行，本书第二部分我们会具体介绍。
- SAP HANA Client：是一系列的库文件，供其他应用程序访问 SAP HANA 数据，库文件包含 JDBC、ODBC、ODBO 等(仅在 Windows 上使用)。

由于索引伺服器是 SAP HANA 最主要的组件，因此我们有必要单独对其进行详细介绍。



- 进程管理组件：用于创建和管理客户端对 SAP HANA 系统的连接和推进，连接创建后，客户端就可以使用 SQL 语言来和索引伺服器进行沟通。
- 交易管理组件：用来协调管理数据库事务，监控事务的开启、运行和关闭。交易管理组件和 SAP HANA 持久层一起确保事务的原子性和数据持久性。
- 请求处理组件：用来处理客户端发出的请求，比如数据操作语句的请求会被 SQL 引擎处理，数据定义语句的请求会被分配到元数据管理组件处理，交易相关的请求会被交易管理组件处理，计划命令会被分配到计划引擎处理。
- 元数据管理组件：存储管理 SAP HANA 中的数据表、视图、列、索引，存储过程的数据的定义信息。
- 关系引擎：支持管理列存储和行存储相关的关系操作。
- 持久层和存储：负责数据的持久性和原子性。使用日志、页面、存储点确保提交过的数据在系统断电或重启后还可以恢复。
- 权限管理组件：当连接建立时，身份验证(Authentication)就会介入，比如要求输入用户名和密码。身份验证组件主要负责用户有适当的权限去执行操作。在第二部分，会有详细章节介绍用户和权限管理。

第三节　SAP HANA 应用场景

目前，SAP HANA 支持如图 2-3 所示的应用场景。

- 第一类场景是单纯用做数据库，和其他关系型数据库一样。
- 第二类场景是用做数据仓库，建立数据集市。
- 第三类场景是作为平台让合作伙伴(OEM)或者独立软件开发商(ISV，Independent Software Vendor)开发交付基于 SAP HANA 的应用。
- 第四类场景是用在应用加速器(Accelerators)，比如我们常说的 CO-PA Accelerator 和 SAP ERP Operational Reporting Accelerator 等。
- 第五类场景是用于新开发的打包应用(APPs)，比如 SAP 推出的基于 SAP HANA 的 Smart Meter Analytics 和 Cash and Liquidity Management 等。
- 除此之外，SAP HANA 也是 SAP 现在产品的基础架构，比如 SAP ERP 和 SAP Business One 等。

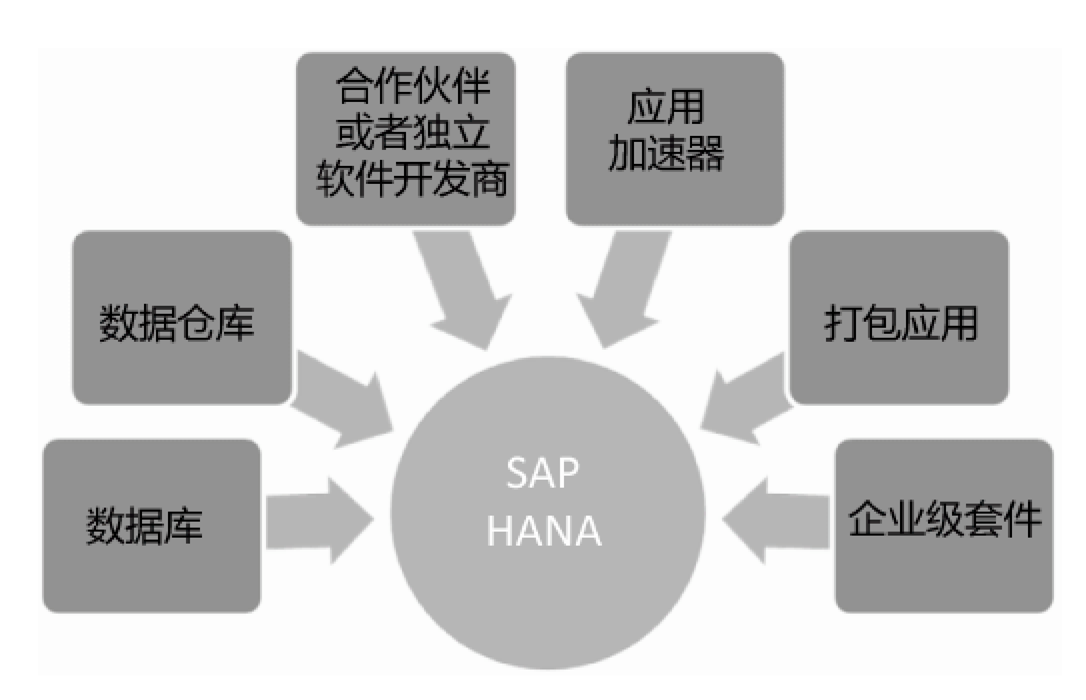


图 2-3

如果从技术架构角度来说，则现在有的 SAP HANA 使用场景包括如下：

一、并排场景(Side by Side Scenarios)

- HANA platform(Datamart)：将不同数据源汇入 SAP HANA 建立数据集市，用 SAP BO BI 做数据展示。
- HANA Apps for Suite：将 SAP Business Suite 的数据导入 SAP HANA，然后在 SAP HANA 基础之上开发应用，比如 ERP Operational Reporting 和 Sales Analysis for Retail 等应用。
- HANA Accelerators：SAP 发布的一些可快速部署的基于 SAP HANA 的加速器，比如 COPA。

所谓并排场景，意思是说 SAP HANA 数据库和原始应用数据库并行且同时存在。在这种情况下，原始应用主要还是运行在传统的主数据库之上，而 SAP HANA 内存数据库是作为辅助数据库存在的，主数据库上产生的数据会被及时同步到 SAP HANA 内存数据库里用于实时分析和测绘，而 SAP HANA 内存数据库产生的新数据不会被返回写入主数据库，因此数据是单向流动的。

二、集成场景(Integrated Scenarios)

- Cloud on HANA
- BW on HANA
- Business One on HANA
- Business Suite on HANA

所谓集成场景，意思就是说将以前运行在非 SAP HANA 内存数据库的原始应用与 SAP HANA 内存数据库进行集成并优化，例如以前 SAP 的 BW 产品、Business One 和 Business Suite 产品等。在这种场景下，SAP HANA 内存数据库将作为主数据库来运行。

三、创新场景(New Frontiers)

所谓创新场景，是指基于 SAP HANA 平台开发的全新应用与 SAP HANA 内存数据库集成，这些新应用能够最大化利用 SAP HANA 内存数据库的实时快速特性来为企业的决策分析提供完全不同的体验。

第四节　SAP HANA 应用开发

在本小节中，我们将简单介绍下基于 SAP HANA 平台的应用开发原理，对 SAP HANA 应用开发有兴趣的读者，则可以在下一部分中了解更多关于 SAP HANA 应用开发的知识。

对于应用开发，大家可能对经典的 MVC 构架，即模型(Model)、视图(View)、控制器(Controller)已经很熟悉，图 2-4 是传统的有应用服务器层的三层应用架构。

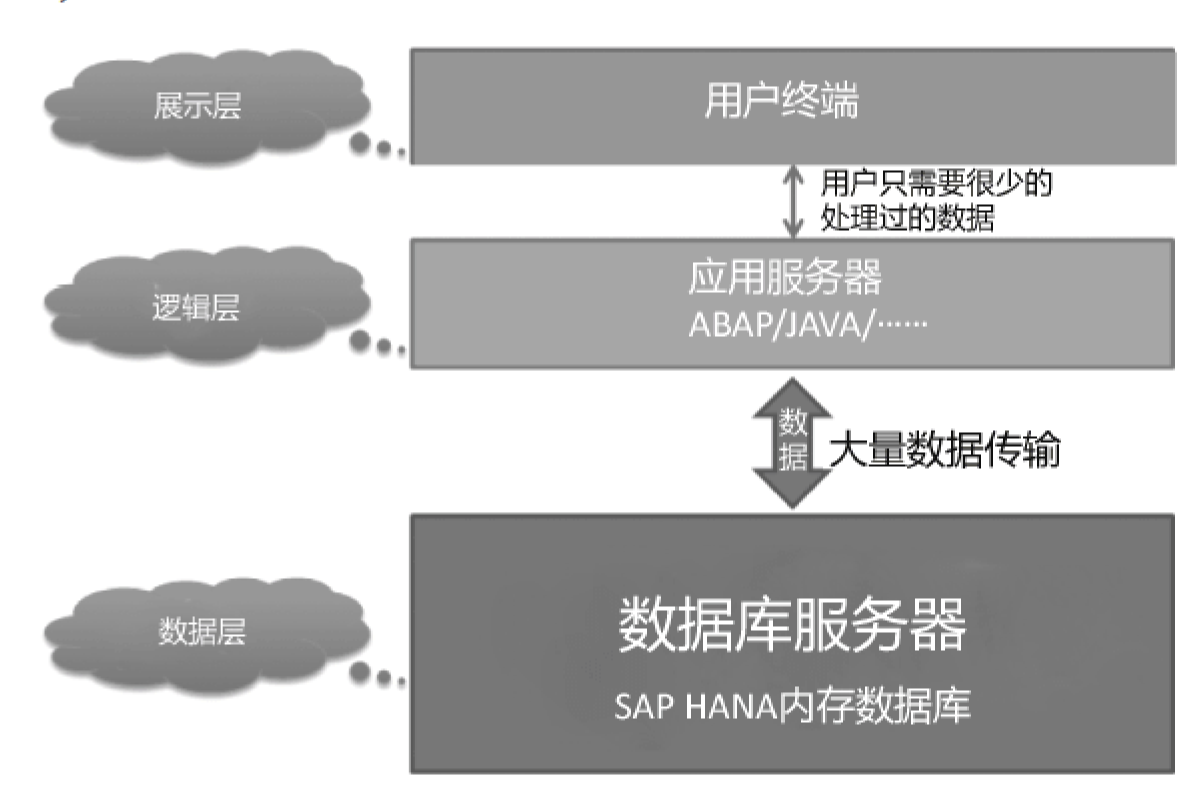


图 2-4

这种传统的 MVC 构架下的应用常见于 SAP HANA 集成场景，即 SAP HANA

只作为内存数据库为应用提供运行所需要的数据。在这种构架下，应用是无法充分利用 SAP HANA 平台的优势的。

从 SPS05 开始，SAP HANA 推出了基于 SAP HANA XS 的原生应用开发方式。所谓原生，就是应用逻辑层也在 HANA 构造和执行。如图 2-5 所示，也就是将应用服务器嵌入 SAP HANA，业务逻辑层下沉，和数据层同处于一个层次，数据层和应用层紧密集成绑定。

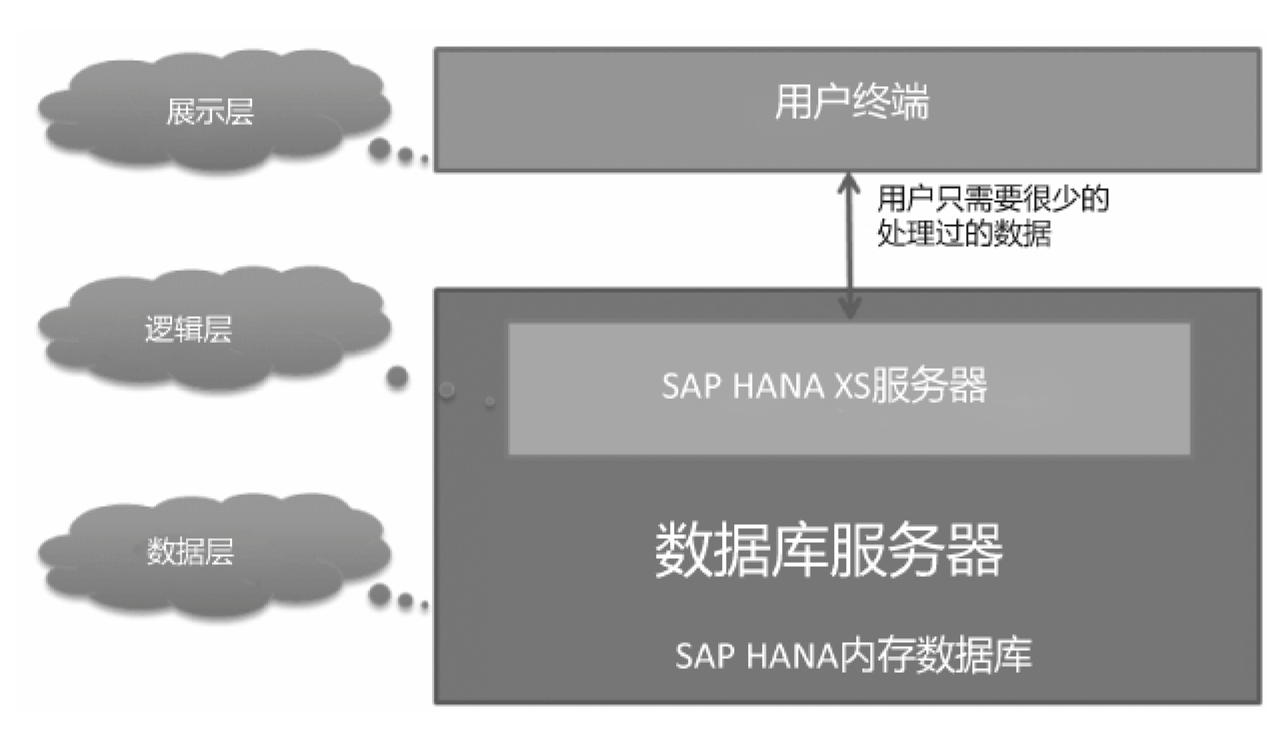


图 2-5

基于以上两点，我们可以将基于 SAP HANA 应用开发分为两类(见图 2-6)。

- 一类是基于 SAP HANA XS 开发的原生应用，业务逻辑层和数据层都在 SAP HANA 中实现。
- 另一类是传统的三层应用开发，应用层需要通过 JDBC/ODBC 读取 SAP HANA 数据。

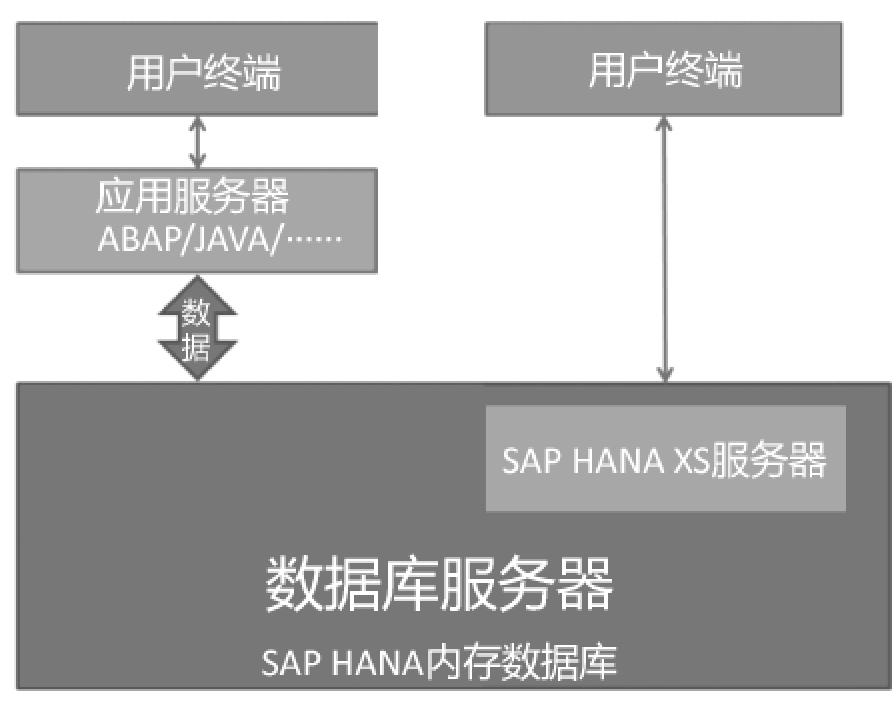


图 2-6

对于传统的三层应用开发，本书将不做具体介绍。在本书第二部分中我们将着重介绍基于 SAP HANA XS 的原生应用开发。

第五节　SAP HANA 企业云

当下，“云”的概念已经越来越深入人心。在这一节，我们将会介绍 SAP HANA 企业云(SAP HANA Enterprise Cloud)。

根据 IDC 公司调查，到 2016 年 70%的企业技术总监会选择云优先策略。很多人可能已经享受了很多云服务，却并不知觉或对什么是云仍是“云里雾里”。百度百科对云的解释是，“广义云计算指服务的交付和使用模式，指通过网络以按需、易扩展的方式获得所需服务。这种服务可以是 IT 和软件、互联网相关，也可是其他服务。”当然，这仍是一个比较抽象的定义。SAP 在云的三个层次上也提供了相应的产品和服务：

- Software-as-a-Service(SaaS，软件即服务)：运行在云计算基础设施上的应用程序，如 SAP 的人力资源管理云计算软件 SuccessFactors。
- Platform-as-a-Service(PaaS，平台即服务)：供客户或企业开发自己基于云的应用，如 SAP HANA Cloud 平台。
- Infrastructure-as-a-Service(IaaS，基础设施即服务)：提供给客户的对所有设施的利用，包括处理、存储、网络和其他基本的计算资源，如 SAP HANA 企业云。

下面我们来着重介绍下 SAP HANA 企业云，可能会有读者问，什么是 SAP HANA 企业云？

SAP HANA 的企业云是一种结合了管理服务(Managed Service)的实时性全方位云平台和云基础设施服务(见图 2-7)。客户可以在一个托管的云环境下，运行基于 SAP HANA 的相关应用程序，包括基于 SAP HANA 的商务套件与 SAP NetWeaver Business Warehouse 等应用。它实现了内存技术实时性与云计算简约性的完美结合，帮助用户快速实现价值。

简单来说，客户只需选定自己需要的企业应用(现阶段所有应用都必须基于 SAP HANA 平台)，而 SAP HANA 企业云服务则会管理运营剩下的所有事情。到目前为



止，在 SAP HANA 企业云中已有多于 100 个预组装(SAP 快速部署解决方案，服务以及 SAP 最佳业务实践内容)的方案可供选择，客户可以在指定时间内快速获得各种业务的启动模板。

由图 2-7 可见，除了已有的技术与应用平台，SAP HANA 企业云还提供了对于客户部署到云的管理服务。



图 2-7

图 2-8 包含了从项目启动到维护运营所需要的完整的生命周期服务，具体包括：

- 评估服务，给客户提供迁移到 SAP HANA 企业云的应用场景、利益与成本分析、所需时间以及可能的风险评估。
- 启动与迁移服务，对客户已有系统提供低风险的迁移方案。
- 云基础服务，提供平台使客户充分享受云上的稳定运行与可预见的 TCO。
- 应用管理服务，在日益灵活的业务环境下，帮助客户利用创新和不断改进的 SAP HANA 企业云服务，满足客户所有的业务需求。



图 2-8



使用 SAP HANA 企业云，客户就不需要操心 IT 相关的所有事务了，可以专注于自身的核心业务。其所带来的好处是显而易见的，例如：

- 成本下降：无需实施费用，降低硬件成本，无需升级费用，无需内部 IT 人员培训费用。

- 时间效益：客户快速获得所需要的各种环境，实施周期可以从以前的几个月甚至几年缩短到几周。
- 很好的灵活性：无需服务器计算能力与机房地点规划，对任何需求都能快速反应，对任何硬件资源都可以进行方便的扩展或收缩。
- 性能提升：使用 SAP 内存技术(HANA)实现更好的性能，通过集成服务可以无缝集成 SAP 或非 SAP 的系统。

获取更多关于 SAP HANA 企业云的最新信息，可以访问 www.saphana.com/cloud。

第三章　SAP HANA 成功案例及实时资料

第一节　SAP HANA 成功案例

一、成功案例——农夫山泉

1. 用户背景

农夫山泉，总部位于浙江省杭州市，是中国领先的矿泉水及饮料的生产商，年销售额近百亿元。农夫山泉在全国拥有 7 个生产基地，数十家工厂，300 多个办事处和超过 100 万的客户，业务涉及生产、销售、计划、调度、物流、市场营销等。

2. 面临的问题

- 数据展现的速度越来越慢，导致了管理效率的降低；
- 数据运算速度随数据量的增大也越来越慢，前台操作人员无法忍受，妨碍了企业业务的正常运行；
- 数据传输的滞后导致决策也相应滞后。

3. 实施 SAP HANA 后的转变

- 快速数据展现，报表生成速度提升了 25～30 倍，使企业能够及时管理日常业务；
- 基于 SAP HANA 强大的内存计算能力，使得针对大量数据的复杂逻辑计算速度提升了 200～300 倍(215 秒→1.8 秒)，极大地提高了企业运营效率；



- 支持超过 8000 名业务人员实时上载资讯，实时数据同步，没有延时(24 小时→实时)。因此，企业能够基于这些数据做实时分析预测，来制定快速灵活的决策。

SAP HANA 在农夫山泉的成功实施，卓越地提升了农夫山泉的商务智能系统，实现了数据展现速度上质的飞跃，实时的数据同步以及高效逻辑计算为企业带来可观的商业价值。

二、成功案例——巴斯夫

1. 用户背景

巴斯夫股份公司(BASF SE)，是一家德国的化学公司，也是世界最大的化工企业。巴斯夫集团在欧洲、亚洲、南美洲和北美洲的 41 个国家拥有超过 160 家全资子公司或者合资公司。公司总部位于莱茵河畔的路德维希港，是世界上工厂面积最大的化学产品基地。巴斯夫在国外的企业大部分在欧洲，几乎遍布欧洲所有的国家。此外，在美国、日本、阿根廷、印度、新加坡、埃及、中国等也都设有分公司或分厂。2012 年，巴斯夫的销售额达 721.29 亿欧元，在全球拥有超过 111 000 名员工。公司业务包括：植保剂、医药、保健品及营养、染料及整理剂、化学品、塑料及纤维、石油及天然气。

2. 面临的问题



巴斯夫拥有 25 万种产品以及超过 40 万名客户。销售人员每天面对的最棘手的问题就是如何实时获知基于产品和客户(25 万×40 万)水平上的分配成本和计算息税前利润(EBIT)数据。

3. 实施 SAP HANA 后的转变

- 能在数秒内分析大量详尽数据，用来为销售人员提供在客户和产品水平上的分配成本和计算息税前利润(EBIT)，这个结果比原来快了 120 倍。不仅如此，快速的分析结果能大大提高数据的可靠性，使企业能够得到及时管理；
- 在需要的情况下，SAP HANA 可以逐笔对业务进行快速分析，让企业在正确的时间生产正确的产品，极大地提高了企业的运营效率；
- 数据实时同步，没有延迟(24 小时→实时)。

三、成功案例——联想

1. 用户背景

联想是一家营业额近 300 亿美元的个人科技产品公司，客户遍布全球 160 多个国家。其为全球第二大个人电脑厂商，名列《财富》世界 500 强，是全球前四大电脑厂商中增长最快的。自 1997 年起，联想一直蝉联中国国内市场销量第一名，占中国个人电脑市场超过三成份额。目前联想在全球拥有员工约 27 000 名，有一流的世界级研发团队分驻于日本大和研究所，中国北京、上海、深圳及美国北卡罗来纳州的罗利市。

2. 面临的问题

- 迫切需要从 CRM 海量数据中得到实时报表来满足业务需要；
- 当前的系统，从数据生成到把最终数据交给使用者需要 8 个小时；
- 无法为业务部提供实时信息。

3. 实施 SAP HANA 后的转变

- 报表生成速度提升了近 1000 倍；
- 数据能实时从 CRM 系统同步到 SAP NANA 系统，并立即展现到使用者面前进行分析；
- 联想内部测试表明，SAP HANA 可以在不到 1 秒钟的时间内处理 180 万条多属性的合同记录。

第二节　获取 SAP HANA 的最新资料

SAP HANA 是一个全新的、开放性的平台，并且还处于快速发展中，正因为如此，如何能让广大客户和开发者实时了解 SAP HANA 的进展，取得相关的资料已经成为 SAP 公司非常重视的一个问题。SAP 公司为了打造一个良好的 SAP HANA 开发生态圈，已经在多处发布资料以满足来自客户和开发者的需要，接下来，让我们来看看如何能得到这些资料吧。



一、最权威的官方文档集合地：http://help.sap.com/hana

在这里你可以找到和 SAP HANA 相关的所有官方文档，包括如何安装 SAP HANA，如何配置 SAP HANA 客户端，SAP HANA 开发者手册，SAP HANA SQL 参考等非常有用的重量级文档。这些文档随着 SAP HANA 每次版本更新也在不断更新中。我们不推荐你把文档下载到本机，因为这样当在线文档更新时你会错过知道更新内容的机会。SAP HANA 还在快速发展中，因此养成在线浏览的习惯会帮助你随时掌握 SAP HANA 的更新进程。

二、SAP HANA 官方网站：http://www.saphana.com/welcome

这是 SAP 公司专门为 SAP HANA 开发的网站，在这里你能了解一切有关 SAP HANA 的最新信息，同时这里也有很多学习的资料和教学视频，最关键的是，这里面的所有资料，包括视频教学都是免费的。

我们强烈推荐你多学习“Learn”和“Try”两个菜单栏里面的内容。“Learn”里面包含了关于 SAP HANA 各个方面知识的资本资料，并且这些资料根据 SAP HANA 的不断更新而时刻保持更新。“Try”里面则包含了 SAP HANA 的试驾系统和演示案例，目前你可以亲自操作 6 个试驾系统来感受 SAP HANA 非同凡响的操作体验，同时你还能找到非常多的在线演示的案例来对 SAP HANA 做进一步的了解。

与此同时，很多的 SAP HANA 专家会在“Blog”里面发布很多他们关于 SAP HANA 某个特定题目的博文，每篇博文都写得很精彩，值得认真阅读。有时间的话我们推荐你多看看多参考练习，你的自身水平将会有很大的提高。



三、SAP HANA 开发者中心：http://scn.sap.com/community/developer-center/hana

这是最著名的关于 SAP HANA 的社区，全球的 SAP HANA 开发者都聚集在此，形成了一个非常好的讨论氛围。你有任何与 SAP HANA 相关的问题都可以在此提出，然后等待来自 SAP 的资深专家或者其他第三方开发者为你解疑释惑。假如你有心得想要分享，这里也是一个不错的地方，你可以建立日志来分享你的心得，没准还能赢得一众粉丝。我们推荐你注册一个 SCN(SAP Community Network)的账户，因为这里面的很多帖子是需要注册用户才能看到的。

可能你会担心自己的英文不好，无法详细描述自己碰到的那些疑难杂症，或者如果你更习惯看中文的帖子，那这个网站绝对是你的福利：http://scn.sap.com/community/chinese/hana。这是 SCN 专门为说中文的 SAP HANA 开发者建立的一个分论坛，在这里你能找到很多的关于 SAP HANA 的中文资料，也能认识很多使用 SAP HANA 的中国朋友。

四、SAP HANA 微刊：http://kan.weibo.com/kan/3444217594965680

假如你有新浪微博的账户(如果没有那就赶快注册一个吧)，那这个微刊你绝对要订阅一下。这本微杂志每周都会发布几篇不错的关于 SAP HANA 的文章，有中文的(大多数)也有英文的。目前这个微刊已经包含了 200 多篇文章，很值得一读。

五、OpenSAP：https://open.sap.com

OpenSAP 是 SAP 公司著名的免费在线培训网站，你可以注册为其中的会员来参与每月不同的在线培训教程，这些教程都是由 SAP 公司资深的构架师或者开发专员亲自备课和讲解的，不过效果非常棒。不过听这些课程的最大前提是要有良好的英文基础。

第二部分

SAP HANA 实践篇

第四章　从实例开始 SAP HANA 之旅

从 SAP HANA SPS06 的版本开始，SAP 为 SAP HANA 应用开发者们提供了一套默认用来做研究和演示用的实例，我们称之为“企业采购管理”(Enterprise Procurement Management)实例。这个实例包含了供应商信息、供应商地址、采购订单和销售订单等大多数人非常熟悉的数据表以及相应的基础数据。这些数据表结构清楚，并且包含的字段也不多，即便你对数据表技术不太在行，也能在快速了解每个表的确切含义的基础上，基于这些数据表的数据来做一些简单的分析和查询。与此同时，该实例还整合了多个用于展示的数据模型和 SAP HANA XS 服务应用，基本上涵盖了本书所涉及的大部分开发内容，因此我们有必要在介绍实际应用开发之前来了解一下这个内置实例。

(1) EPM 实例所在的 Schema 如图 4-1 所示。

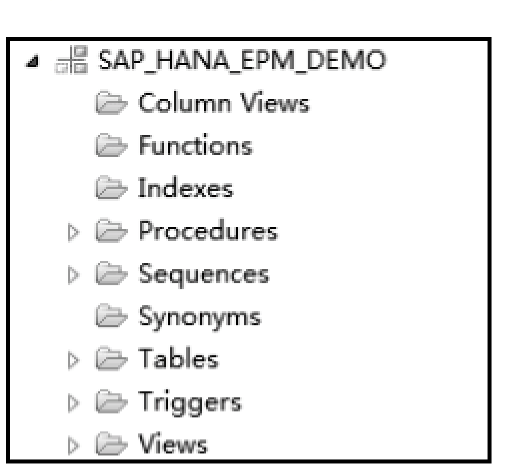


图 4-1

(2) EPM 实例包含的数据表如图 4-2 所示。

sap.hana.democontent.epm.data::addresses
sap.hana.democontent.epm.data::businessPartner
sap.hana.democontent.epm.data::constants
sap.hana.democontent.epm.data::employees
sap.hana.democontent.epm.data::messages
sap.hana.democontent.epm.data::products
sap.hana.democontent.epm.data::purchaseOrder
sap.hana.democontent.epm.data::purchaseOrderItem
sap.hana.democontent.epm.data::salesOrder
sap.hana.democontent.epm.data::salesOrderItem

图 4-2

(3) EPM 实例包含的序列(Sequences)如图 4-3 所示。

sap.hana.democontent.epm.data::addressId
sap.hana.democontent.epm.data::employeeId
sap.hana.democontent.epm.data::partnerId
sap.hana.democontent.epm.data::purchaseOrderId
sap.hana.democontent.epm.data::salesOrderId
sap.hana.democontent.epm.data::textId

图 4-3

(4) EPM 实例所在的包(Package)如图 4-4 所示。

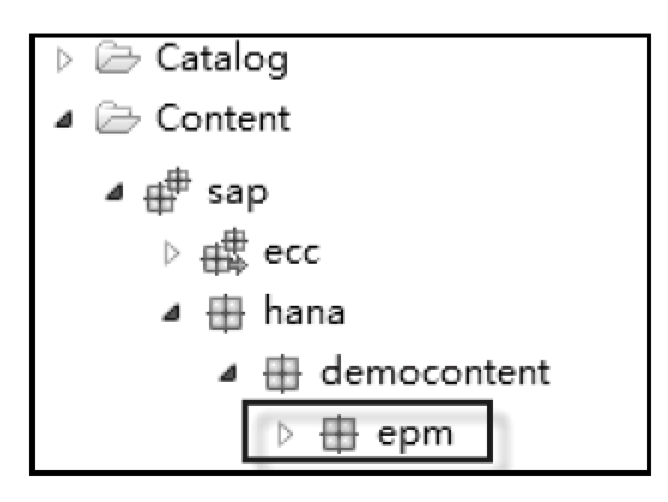


图 4-4

(5) EPM 包含的数据模型位置如图 4-5 所示。

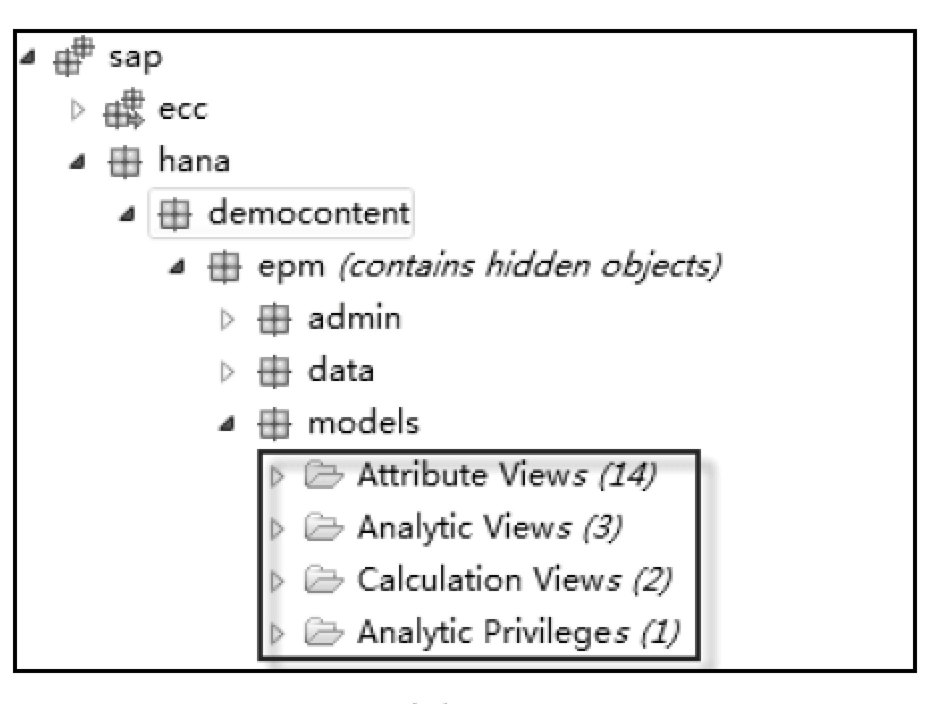


图 4-5

如果你现在看不懂上面这些图片的内容，那很正常，在下一章里面我们会详细介绍如何在 SAP HANA Studio 里面导入这个实例，以及如何检查实例中的数据表和用做演示的数据模型，如何用实例自带的工具生成演示数据等有意思的操作。如果你已经迫不及待，那赶紧翻到下一章，让我们开始 SAP HANA 之旅吧。

第五章　配置 SAP HANA 开发环境

现在终于到了能实际操作 SAP HANA 的章节了。从本章节开始，我们将逐步带领你从配置 SAP HANA 开发环境，到建立数据模型，以及编写常用的 SQL 查询语句，使用 SAP HANA SQL Script 编写可多次使用的存储过程。最后，我们将介绍如何在前台展现数据来完成整体的开发工作。

第一节　申请试用 SAP HANA

一、申请 SCN 用户

在申请试用 SAP HANA 实例之前，你首先需要成为 SCN(SAP Community Network)的用户。如果你已经是 SCN 的成员，那么恭喜你，你可以跳到下一小节直接去申请试用 SAP HANA 实例。如果你还没有成为 SCN 的用户，我们推荐你申请一个，因为随着你对 SCN 了解的不断深入，你会越来越发现其实用性。请跟随如下的操作来完成 SCN 用户的申请。

(1) 首先登录网址 http://scn.sap.com，随后在出现的页面里面单击“Register”按钮(见图 5-1)。

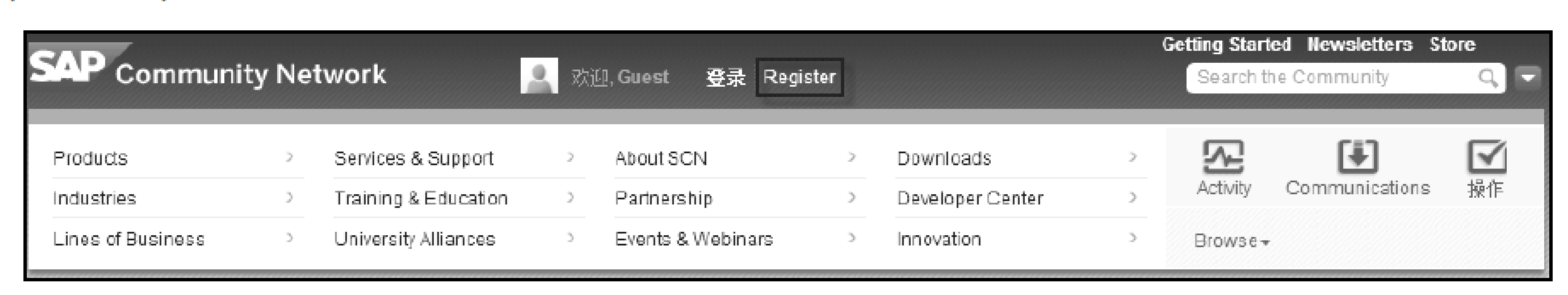


图 5-1

(2) 在弹出页面里面，把带星号(*)的字段都填全，注意你的邮箱地址一定不要填错(见图 5-2)。

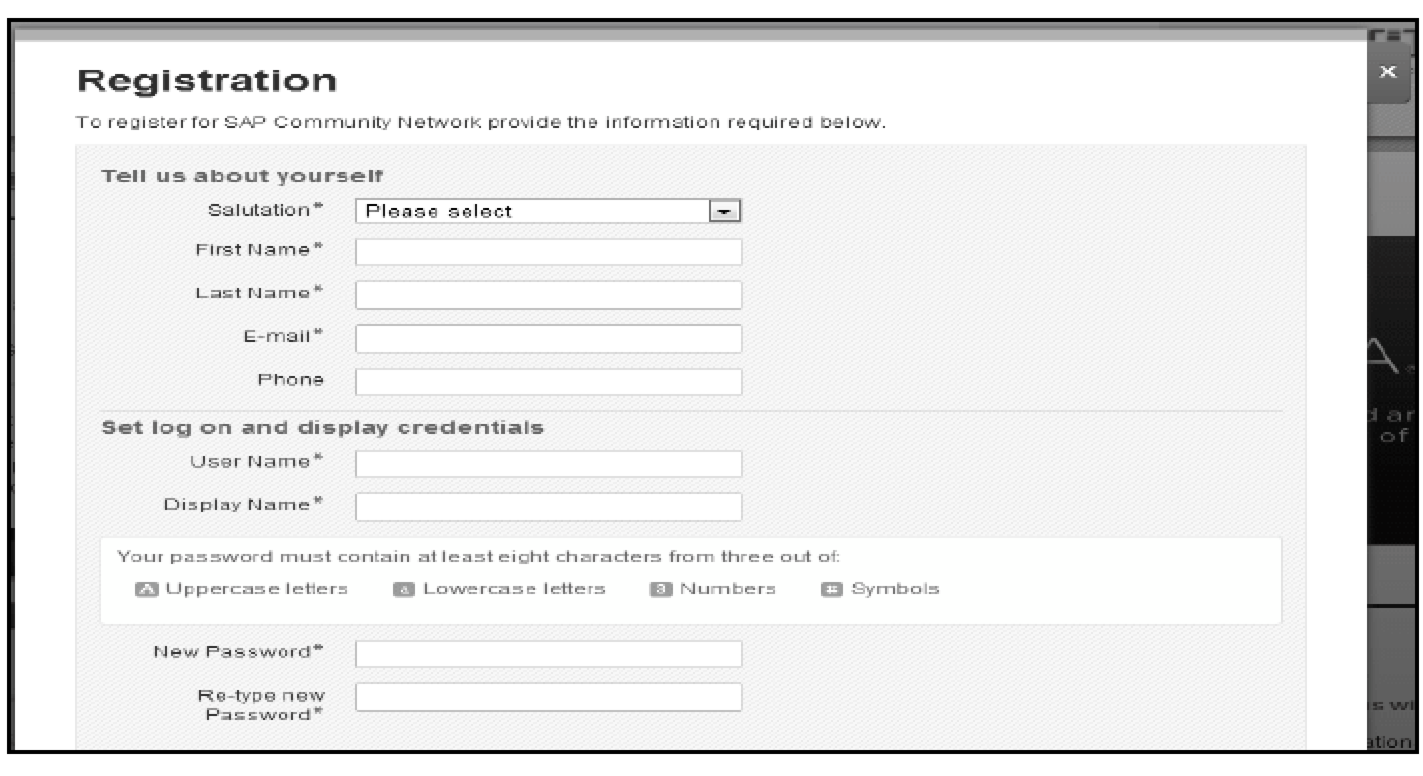


图 5-2

(3) 把信息都填完后，在页面的末尾有相应的 SCN 协议条款，请认真读一下。记得图中圈出的两个单选框都要选中，然后单击“Register”按钮(见图 5-3)。

图 5-3

(4) 系统会通知你注册邮件已发往你的邮箱(见图 5-4)。

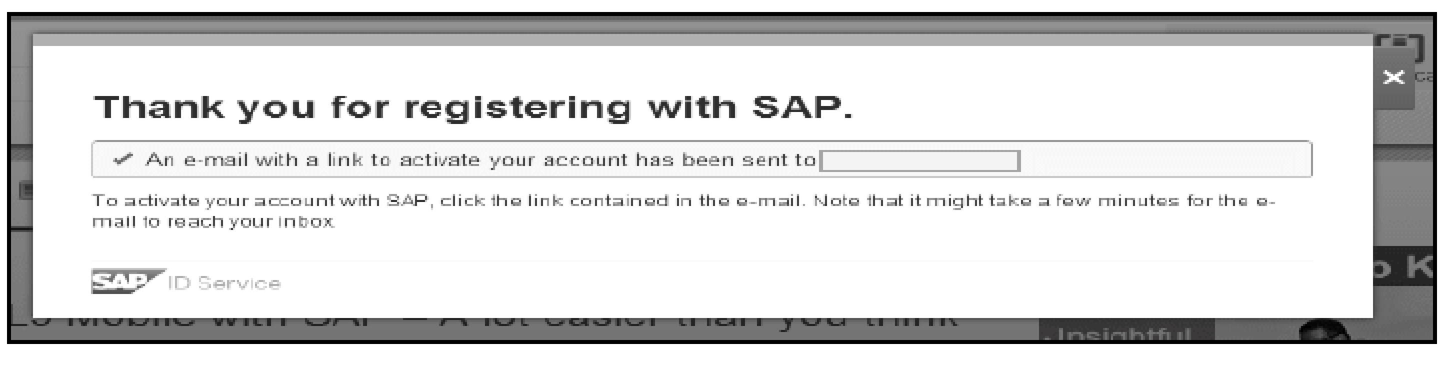


图 5-4

(5) 稍后你的注册邮箱会收到一封来自 SCN 的欢迎信，并邀请你激活你的账户，注册账户被激活后就可以正常登录 SCN 网站了。登录后的界面如图 5-5 所示，你可以通过登录名旁边的下拉菜单来对你自己的资料进行修改或者更新。



图 5-5

二、申请 SAP HANA 开发者实例

首先请在浏览器中打开下面这个网址，在此之前请确保你已经用刚申请的用户登录了 SCN (http://developers.sap.com/hana)。

在 SAP HANA Developer Center 的 Overview 页面里面，你会找到两个如图 5-6 中方框标记出的选项。

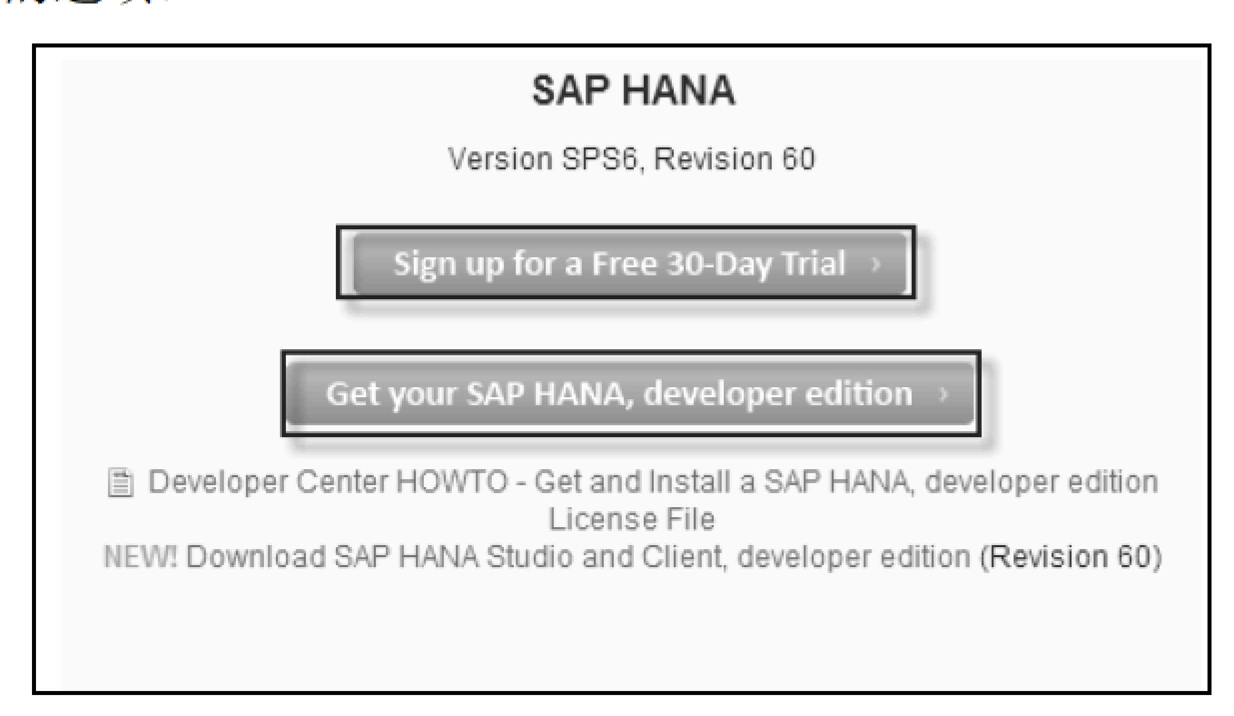


图 5-6

现在简单说说这两个选项的不同。第一个选项“Sign up for a Free 30-Day Trail”可以让你申请一个 30 天免费的 SAP HANA 试用系统。该试用系统由 CloudShare 公司提供云端服务，通过你的 SCN 用户可以直接申请。申请操作非常简单，只需填好申请表格并提交，CloudShare 会自动为你生成两台虚拟机，一台虚拟机运行基于 SUSE Linux 企业版的 SAP HANA SPS 06 Server，一台虚拟机运行基于 Windows 7 的 SAP HANA Studio 以及 SAP HANA Client。与此同时，该虚拟机还安装了你开发所要用的客户端驱动，如 JDBC、ODBC、ODBO、Python，并集成了 SAP Visual Intelligence、SAP BusinessObjects BI、Edge Edition 等常用的软件。除了这两台虚拟



机外，CloudShare 还会为你准备一份用来测试的数据，即 SFLIGHT 数据包。通过这两台虚拟机以及相应的测试数据包，你不用做任何配置就可以开始 SAP HANA 的开发之旅，并可以完成绝大部分 SAP HANA 的开发操作，体验 SAP HANA 的最新特性。但是有一个小小的遗憾，即它只有 30 天的免费期。这里我们需要指出的是，随着 SAP HANA 的版本不断更新，你在页面看到的版本信息以及得到的免费试用的软件信息也许与本书并不完全一致。

因此，在 30 天免费期过后，如果你仍然想要继续 SAP HANA 的开发，你就需要考虑第二个选项“Get your SAP HANA，developer edition”。在这里，你可以找到多家和 SAP 有合作关系的云服务供应商，如 Amazon Web Services、KT ucloud biz，包括我们前面提到的 CloudShare 等，他们提供基于云端的 SAP HANA 基础平台服务，并可以按照你的需求定制你专属的虚拟服务器。因为有多家供应商可供比较，所以你可以选一个硬件看上去还不错、价钱合适的供应商去租赁一台虚拟的服务器，然后继续你的 SAP HANA 开发。在此我们需要声明，对于开发者版本来讲，SAP 是不会对你收取任何关于 SAP HANA 许可证的费用的，你所交的所有费用是云服务商收取的硬件维护费用。如果你开发的产品需要销售，可申请 SAP HANA One Business Edition。

三、安装 SAP HANA Client 以及 SAP HANA Studio

本小节我们将介绍如何在你自己的本机安装 SAP HANA Client 和 SAP HANA Studio 这两个常用的 SAP HANA 开发工具包。首先我们需要取得安装文件，那从哪里能取得最新的 SAP HANA Client 和 SAP HANA Studio 的安装包呢？答案还是在 SAP HANA Developer Center (http://developers.sap.com/hana)。



SAP HANA Client 和 SAP HANA Studio 开发者版本对于普通开发者来说其下载和安装是完全免费的。需要指出的是，SAP 只会在这个网站发布这两个软件包最新的版本，所以请不要随意使用网上流传的一些旧的版本，因为过老的版本很容易在开发过程中出现死机、中断或者无法连接的问题。目前 SAP HANA Studio 最新的版本号是 60，以后当你发现这里的版本号有更新时，请及时下载最新的软件包进行更新。

单击图 5-7 框内的链接，你会进入如图 5-8 所示的界面：



图 5-7

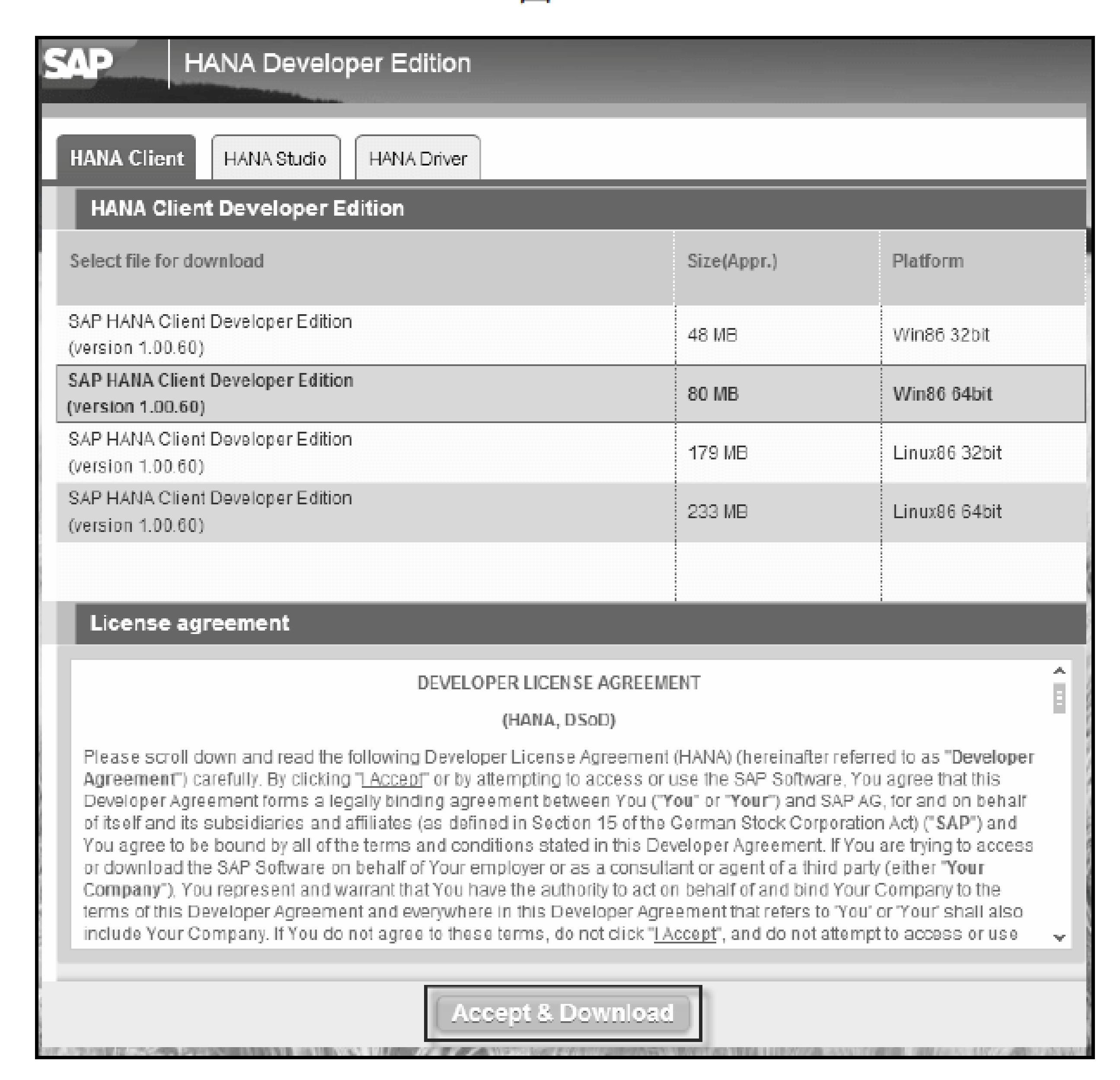


图 5-8

分别进入“HANA Client”和“HANA Studio”标签页，选中适合你操作系统的版本，然后单击“Accept & Download”按钮触发浏览器下载进程。需要注意的是，在你点击该按钮之前，请务必详细阅读“License agreement”文本框内的内容。

下载完成后，在你设定的下载路径下找到安装压缩包并解压，找到安装包里面如图 5-9 所示的安装文件，双击安装即可，因为这些软件是免费的，所以安装过程中安装程序不会向你索要密钥或者许可证。



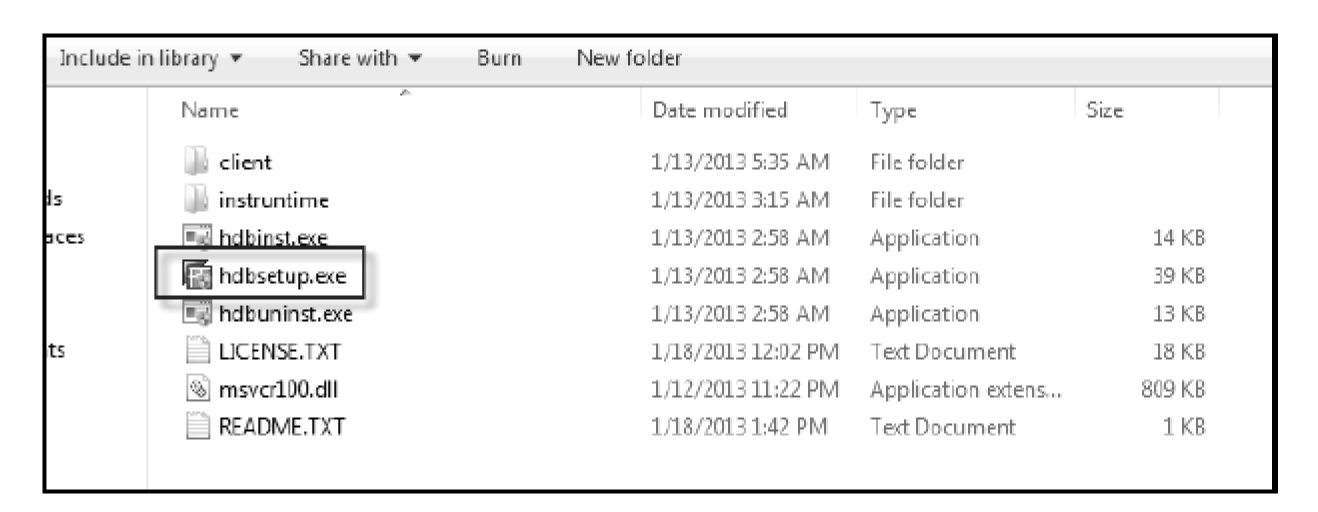



图 5-9

四、连接 SAP HANA Studio 到 SAP HANA

通常来说在你安装完 SAP HANA Studio 以及 SAP HANA Client 后，你会在 Windows 桌面的开始菜单栏里面找到 SAP HANA Studio 的快捷方式，如图 5-10 所示。



图 5-10

单击此快捷方式，启动 SAP HANA Studio。SAP HANA Studio 的初始启动画面如图 5-11 所示。



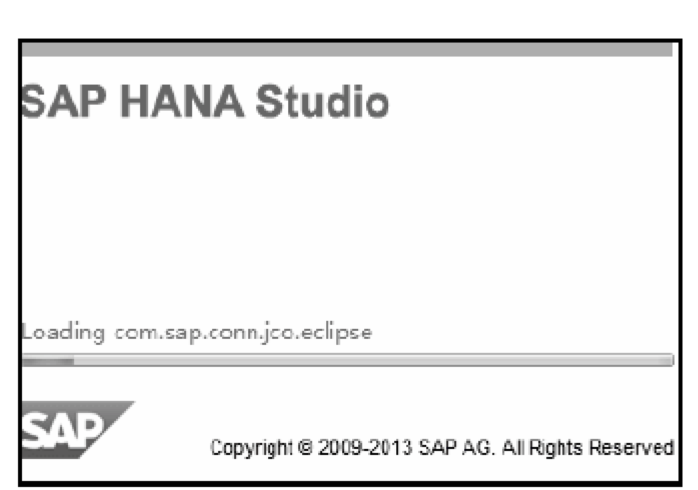


图 5-11

花些时间等待所有的加载项加载完成，一般来说 SAP HANA Studio 的启动会在 1 分钟之内完成。

SAP HANA Studio 是 SAP HANA 的主要开发工具之一。通过 SAP HANA Studio，你可以在 SAP HANA 平台上创建数据模型、存储过程以及各种视图等，或

者操作交互式表格和进行数据表查询。SAP HANA Studio 是基于 Eclipse 开发的，所以如果你有过 Eclipse 或者是 MyEclipse 的使用经验的话，对 SAP HANA Studio 的界面及操作也能够比较快地熟悉。而且正是由于 SAP HANA Studio 是基于 Eclipse 开发的，因此你的操作系统需要安装相应的 Java 运行环境才可以。我们建议你从 Java 中国官方网站(http://www.java.com/zh_CN/)下载最新的 Java Runtime 安装包来进行安装。

第一次打开 SAP HANA Studio，会出现如图 5-12 所示的欢迎界面。

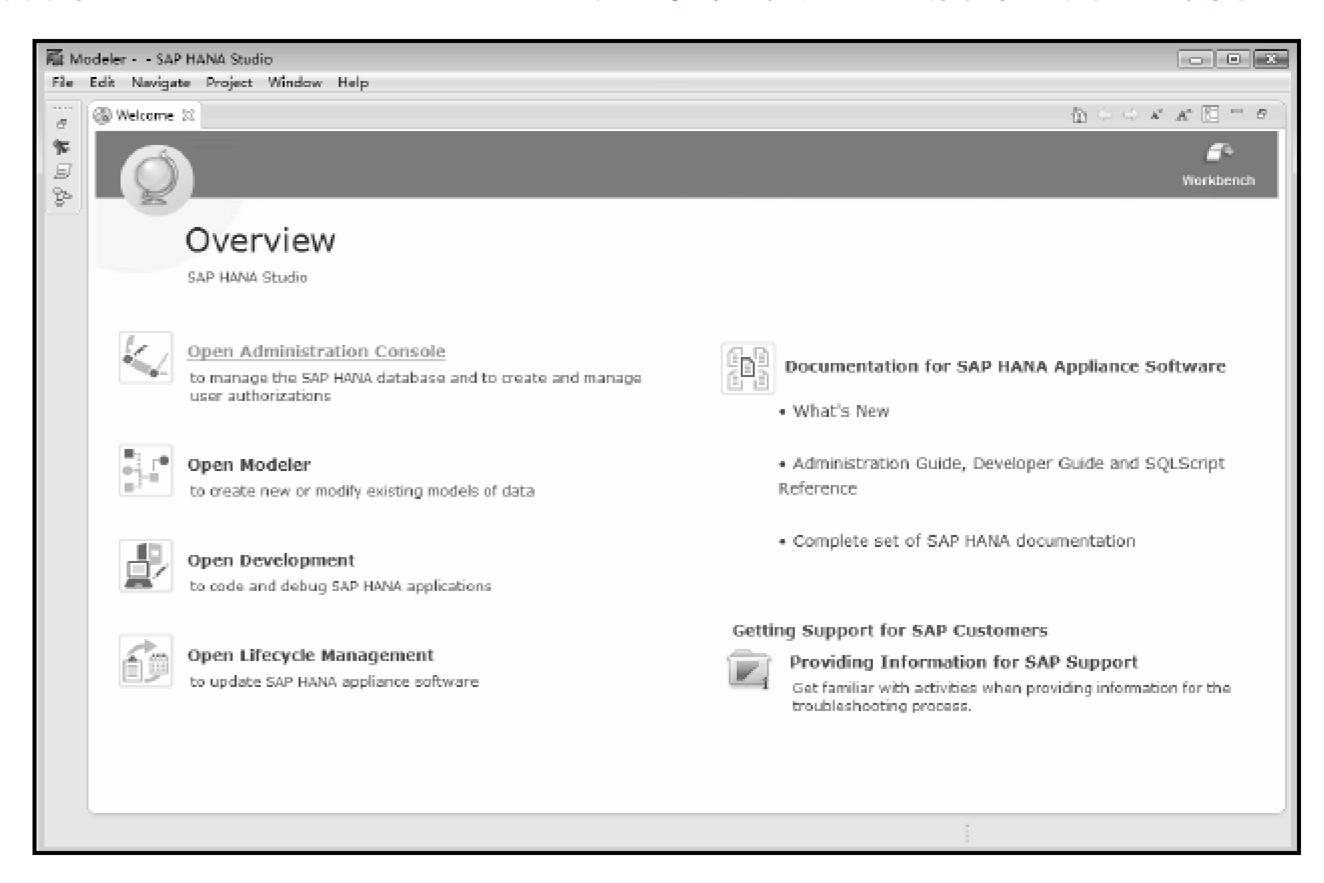


图 5-12

该界面的左边列出了 4 种 SAP HANA Studio 常用的透视图(Perspective)，界面右边则是相关的文档和帮助说明。有关透视图的概念将在接下来的段落里为大家说明，我们现在先要做的是把 SAP HANA Studio 连接到 SAP HANA Serve 上去，具体做法如下：

(1) 单击欢迎界面上的“Workbench”链接(见图 5-13)。



图 5-13

(2) 如图 5-14 所示，我们首先要找到视图“SAP HANA Systems”，在该视图在屏幕左上方的位置，找到后右击此视图的空白处，在弹出的菜单中选择“Add System”一项。

(3) 这时窗口“Systems”会出现，请把你得到的关于你的 SAP HANA Server 的相关信息填入弹出窗口的相关字段中。首先你要从 SAP HANA 系统管理员处得到如下的系统信息：

- 主机地址(Host Name)：本例中我们使用“10.58.5.18”，请根据实际情况替换掉主机地址；
- 实例编号(Instance Number)：本例中我们使用“03”，请根据实际情况替换掉实例编号；
- 用户名(User Name)：本例中我们使用“SYSTEM”，请根据实际情况替换掉用户名；
- 密码(Password)：******。

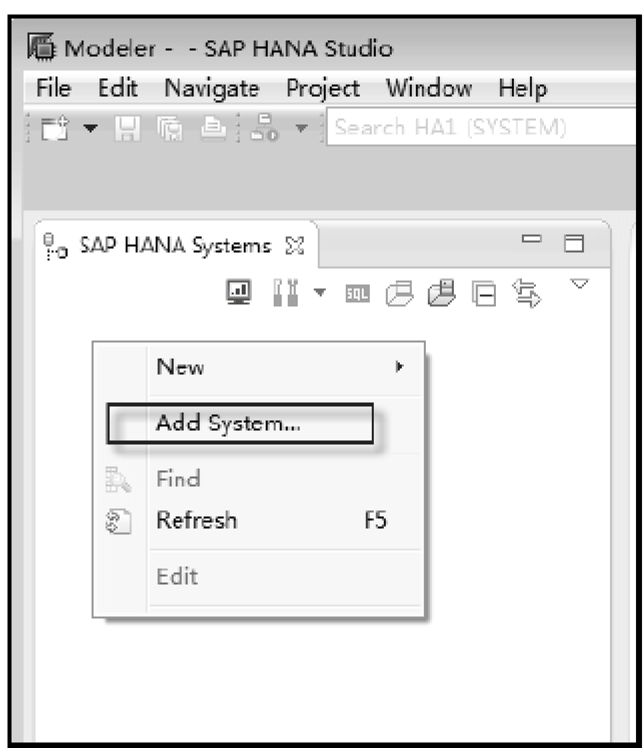


图 5-14

在“System”窗口第一屏要填写主机地址和主机编号及主机描述，最后填好的结果如图 5-15 所示。

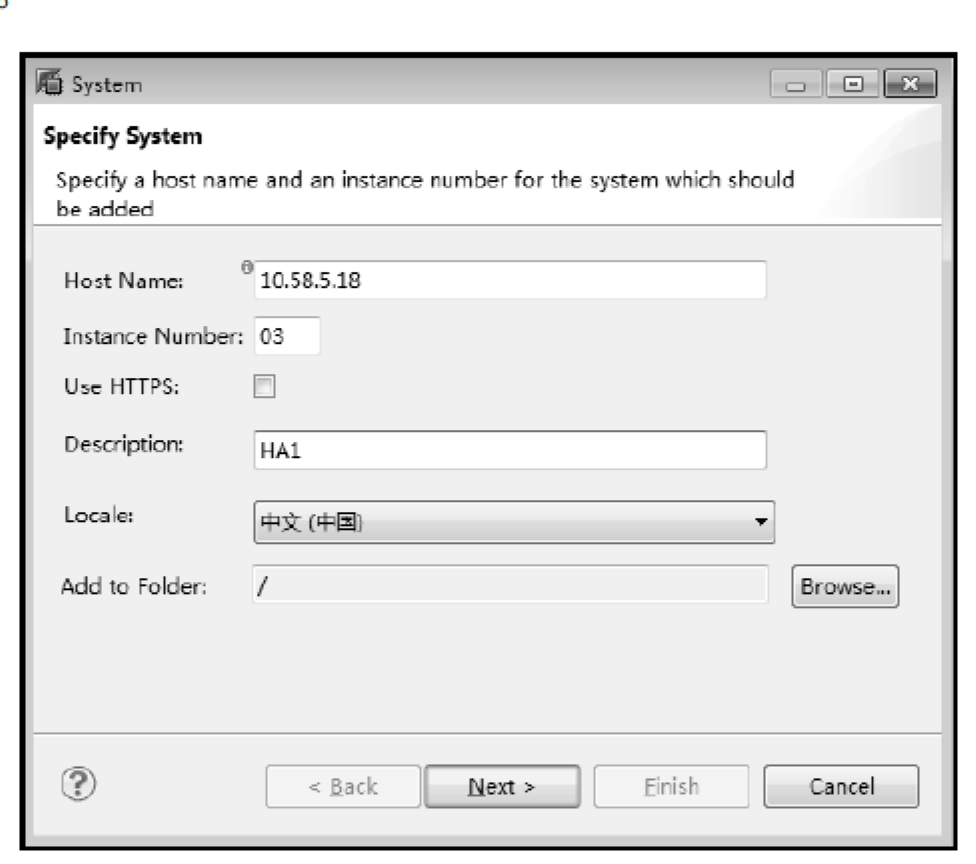


图 5-15

注：如果你的 SAP HANA 主机采用了 HTTPS 传输协议，则要把“Use HTTPS”选项选中。



(4) 单击“Next”按钮，接着输入用户名和密码后单击“Finish”按钮(见图 5-16)。

System
Connection Properties
Specify the properties to be used for connecting to the system.
Authentication can be carried out via the current operating system user or a valid SAP HANA Database user
Authentication by current operating system user
Authentication by database user
User Name: SYSTEM
Password: ••••••••
(To manage your password, see 'Secure Storage')
Connect Using SSL
< Back　Next >　Finish　Cancel

图 5-16

到这里我们就成功把 SAP HANA Studio 连接到 SAP HANA Server 上去了。请按照图 5-17 检查系统图标的右下角是否为绿灯(连接正常)。

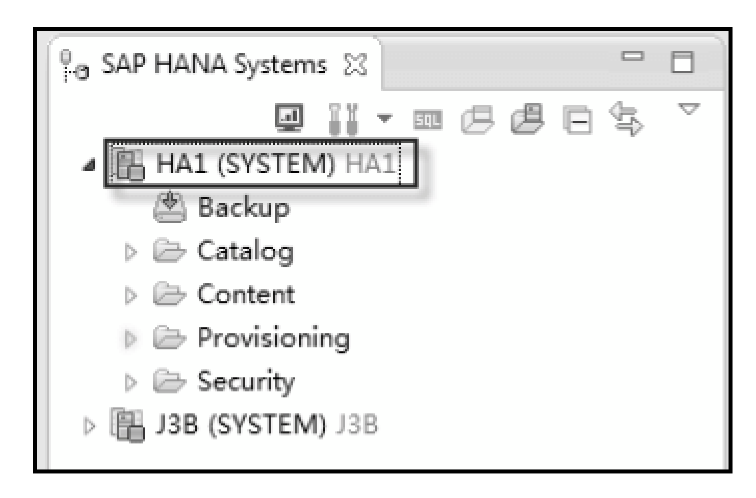


图 5-17

接下来我们介绍下 SAP HANA Studio 里面的视图(View)和透视图(Perspective)。

五、SAP HANA Studio 透视图

所谓 SAP HANA Studio 视图，指的是在 SAP HANA Studio 界面里面能够完成某一特定功能的屏幕元素。比如图 5-18 中列出的“SAP HANA Systems”、“Where-Used List”、“Quick Launch”以及“Properties”都是属于这种类型的屏幕元素。视图是 SAP HANA Studio 最基本的屏幕元素，把这些视图以及它们在屏幕

上排列的位置组合在一起，就形成了 SAP HANA Studio 透视图(Perspectives)。



图 5-18

为了让开发者更加方便地使用 SAP HANA Studio，SAP 把各种常用的视图根据开发情景组合成不同的透视图，通过在不同的透视图之间切换，开发者能够快速展开工作。除此以外，开发者还能在 SAP HANA Studio 中创建自己的透视图，即把自己常用的视图组合起来生成个性化的透视图。

在 SAP HANA Studio 中，通过路径“Window”→“Open Perspective”可以方便地找到 SAP HANA Studio 中默认集成的系统透视图，如图 5-19 所示。

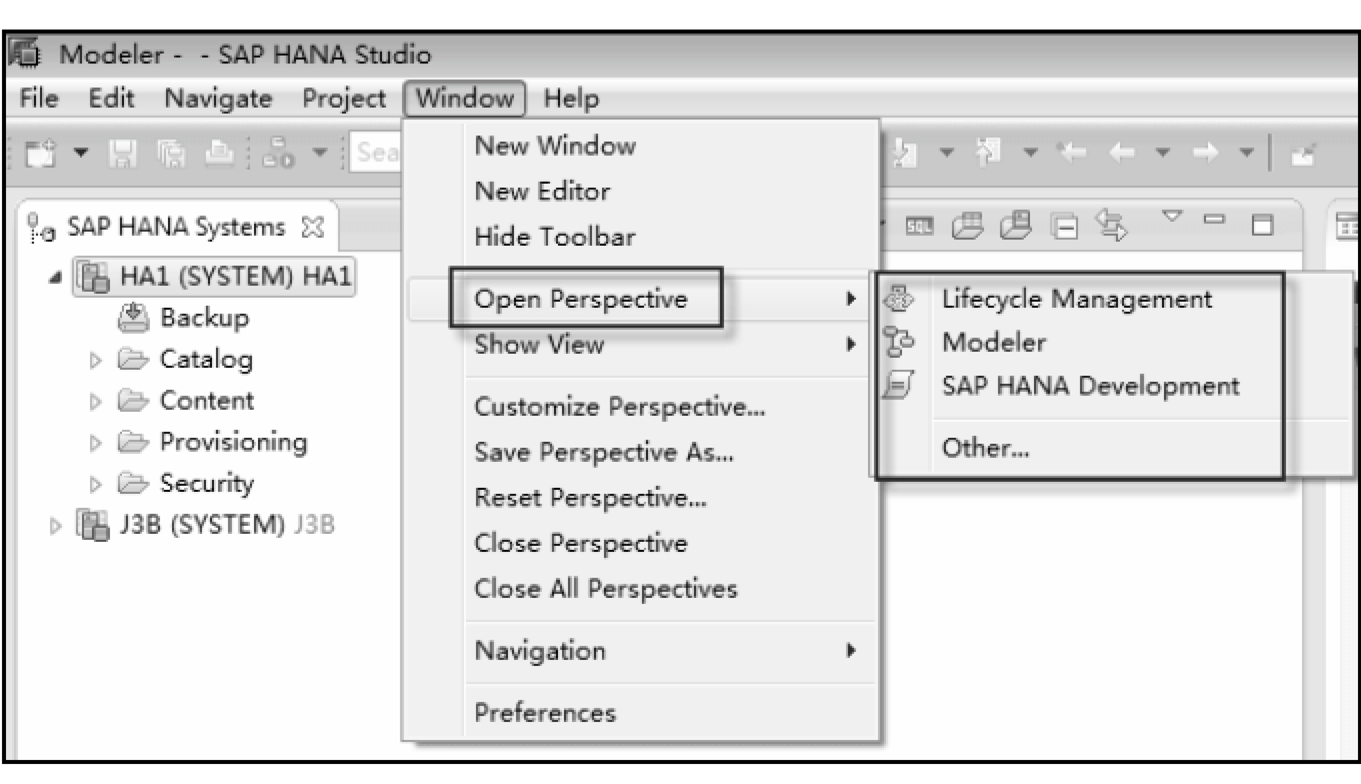


图 5-19

同时在 SAP HANA Studio 的工具栏中，我们也能通过图 5-20 所示的按钮快速打开透视图列表。

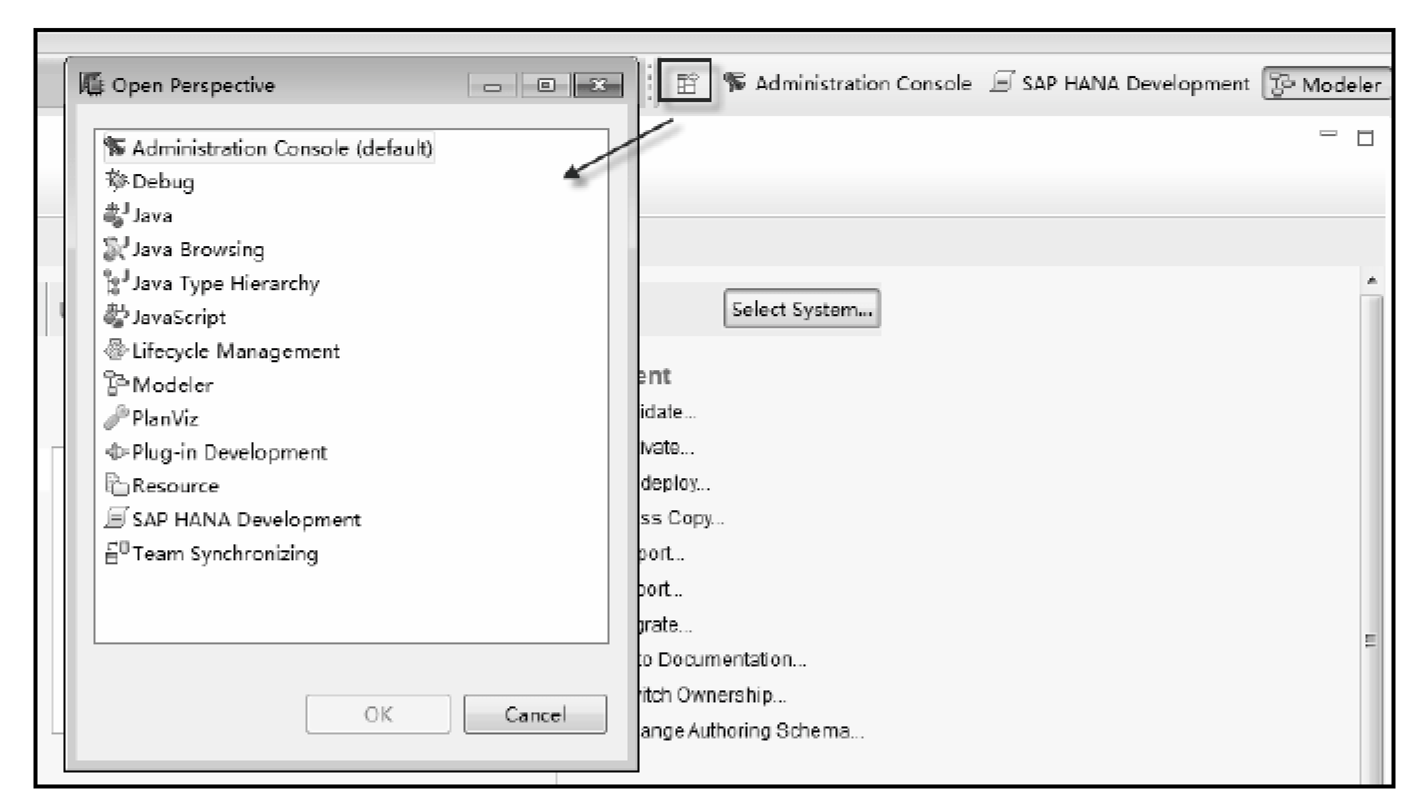


图 5-20

接下来我们简单介绍下 SAP HANA Studio 几个常用的透视图。

1. 建模透视图(Modeler Perspective)

建模透视图通常用来把各种不同的数据视图组合起来以构建分析模型。同时你也可以使用这个透视图来创建包(Package)、存储过程(Procedure)以及交付单元(Delivery Units)。如图 5-21 所示，这个透视图是开发者在 SAP HANA Studio 中最常用的透视图之一。建模透视图主要包含“SAP HANA Systems”和“Quick Launch”两个视图。

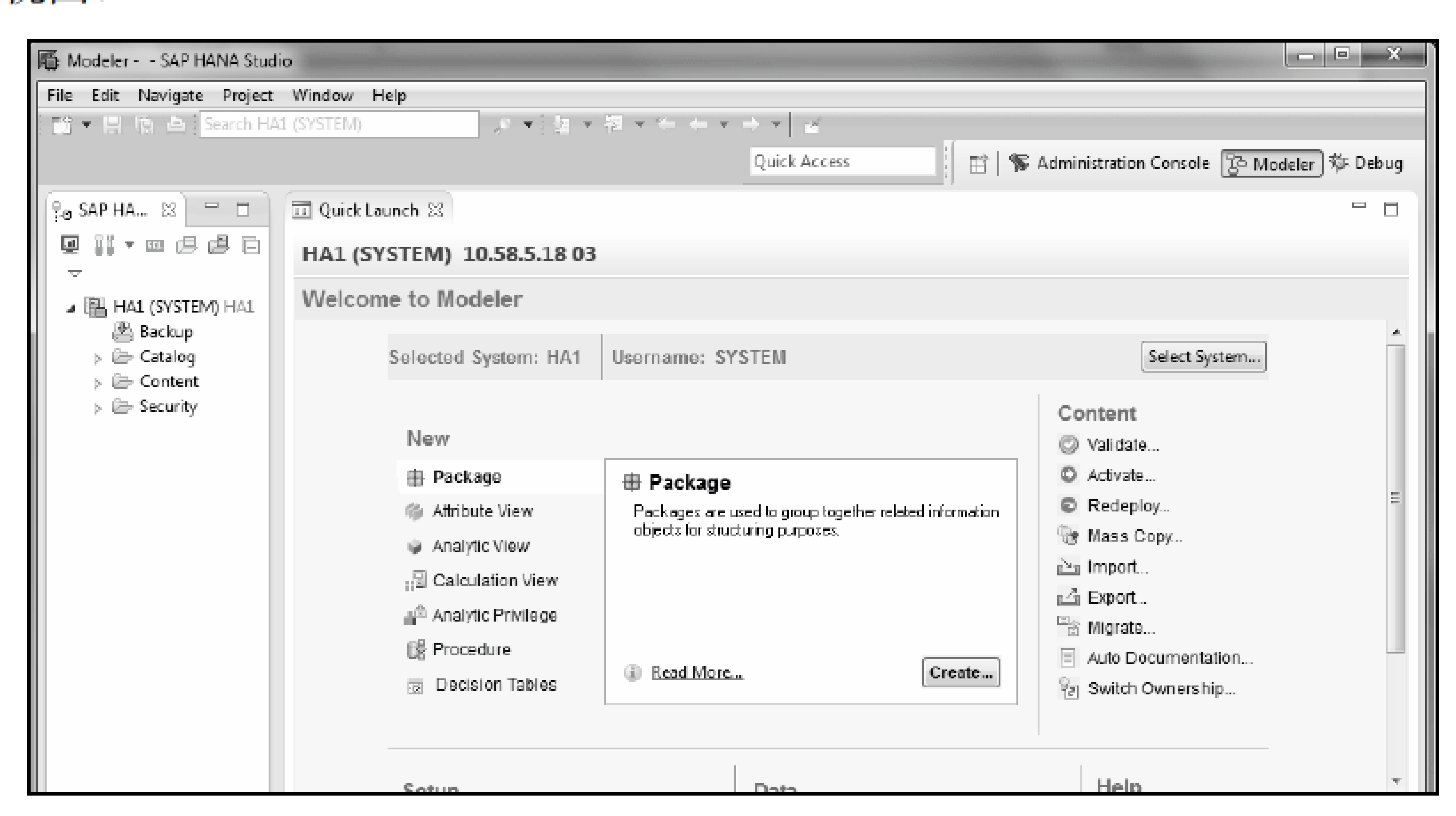


图 5-21



2. 管理控制台(Administration Console Perspective)

SAP HANA 的管理员通过这个透视图来管理、监控以及诊断 SAP HANA 系统的运行情况(见图 5-22)。

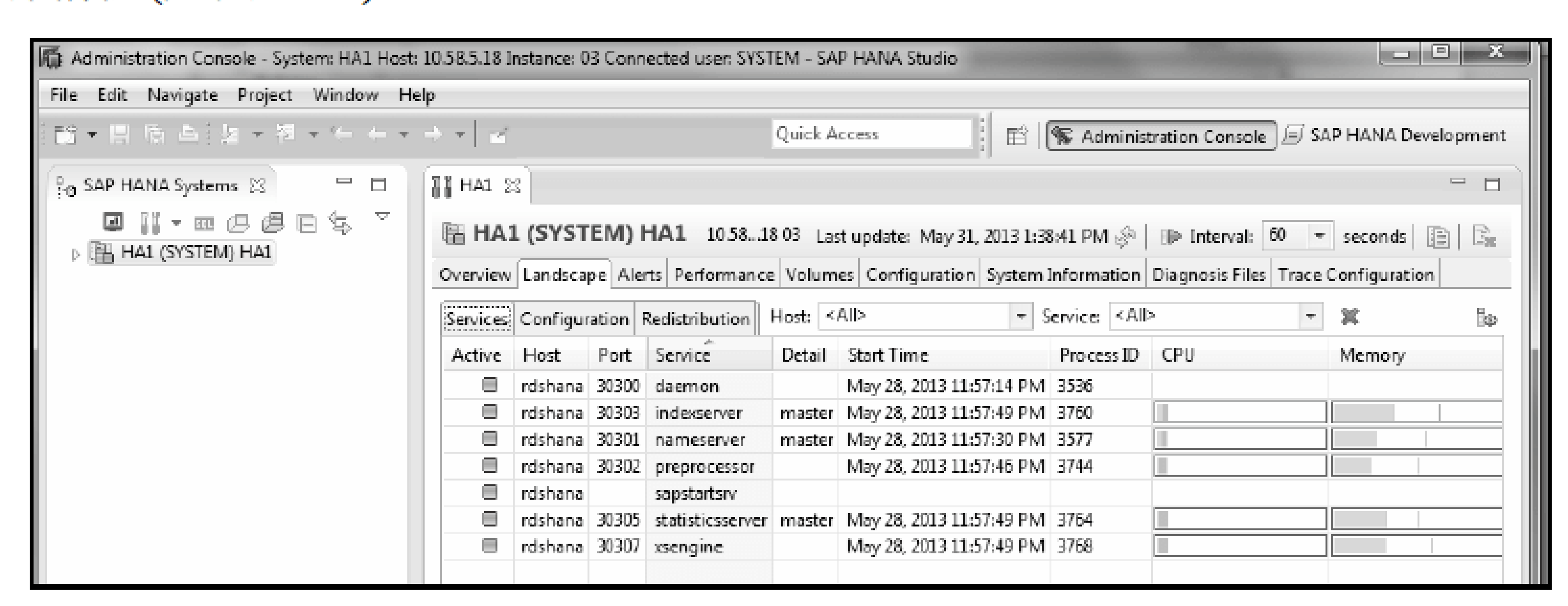


图 5-22

3. SAP HANA 开发透视图(SAP HANA Development Perspective)

顾名思义，SAP HANA 开发透视图是 SAP HANA 开发者的主要工作透视图，它能帮助开发者有效地管理在开发过程中常用的产品组件，如项目(Project)以及开发对象(Development Objects)等。通常来讲，这个透视图由三个主要视图组成，即“Project Explorer”、“SAP HANA Reositories”以及“SAP HANA System”(见图 5-23)。SAP HANA 开发透视图对于今后使用 SAP HANA Studio 基于 SAP HANA 平台的开发工作非常重要，我们将在第九章详细介绍如何通过此透视图来进行一些基本的开发工作。



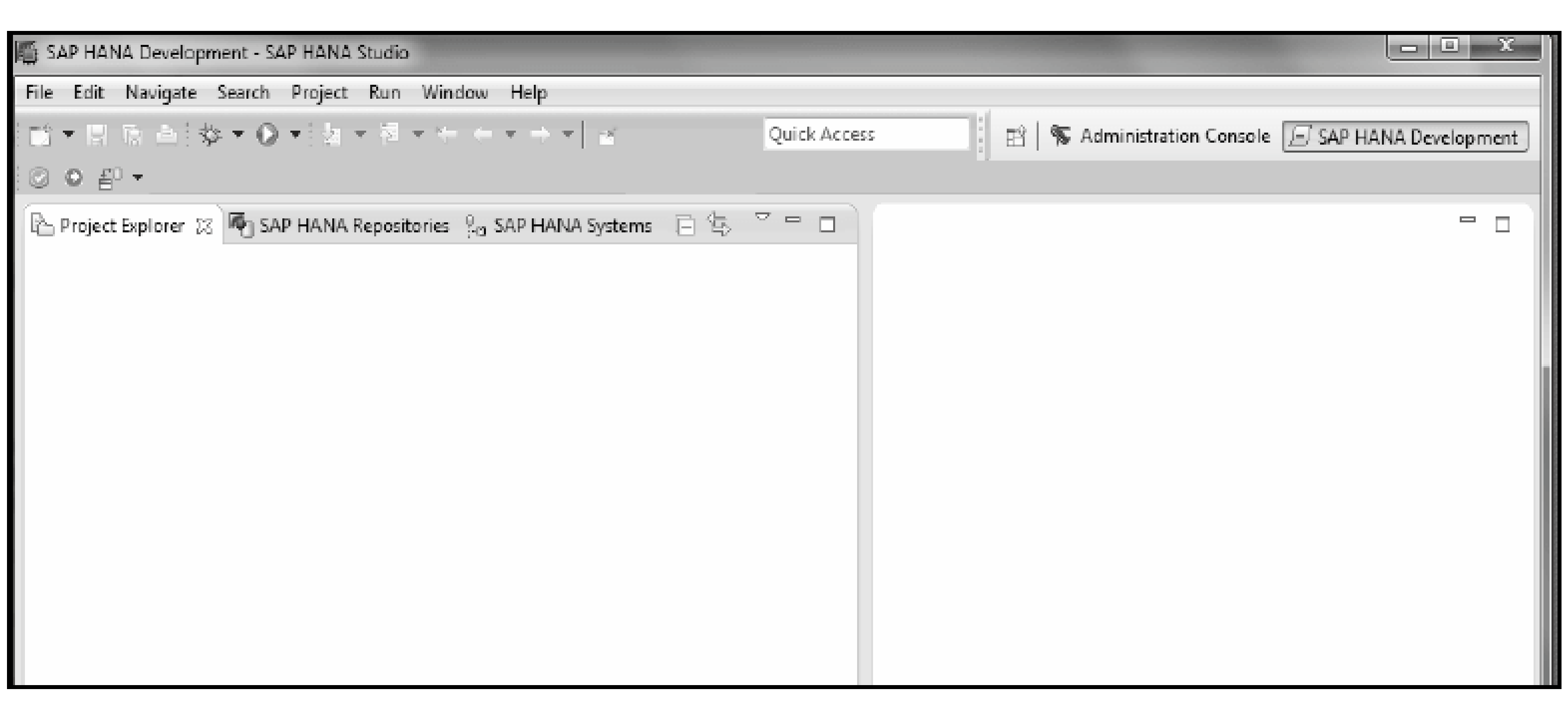


图 5-23

4. 调试透视图(Debug Perspective)

调试透视图可以允许程序开发者在 SAP HANA 系统上调试代码。同时，调试透视图提供的多种工具能够极大地提高开发者调试程序的效率，如图 5-24 所示。

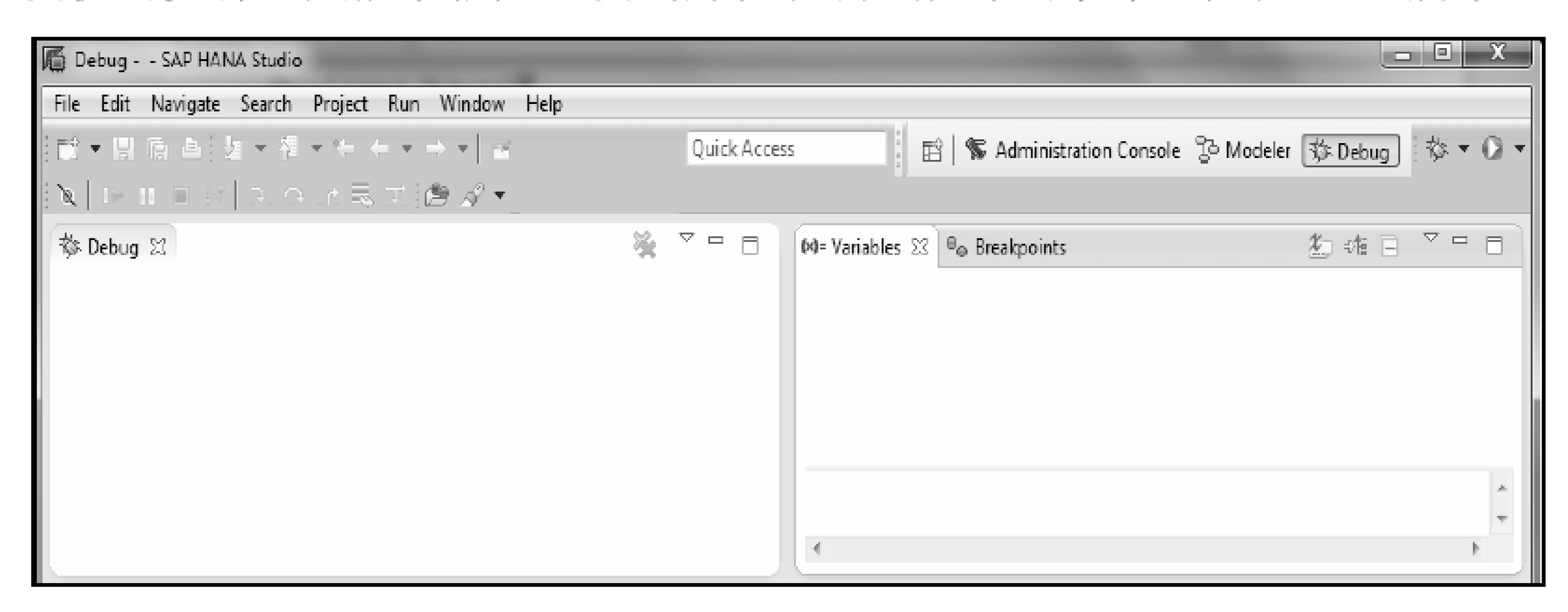


图 5-24

六、自定义透视图

前面我们介绍了很多 SAP HANA Studio 自带的透视图，你可能会问，那如何定制属于自己的透视图呢？请跟随以下的步骤来生成你自己的透视图吧。

(1) 在工具栏里面定位到“Window”→“Close All Perspective”来关闭所有已经打开的透视图(见图 5-25)。

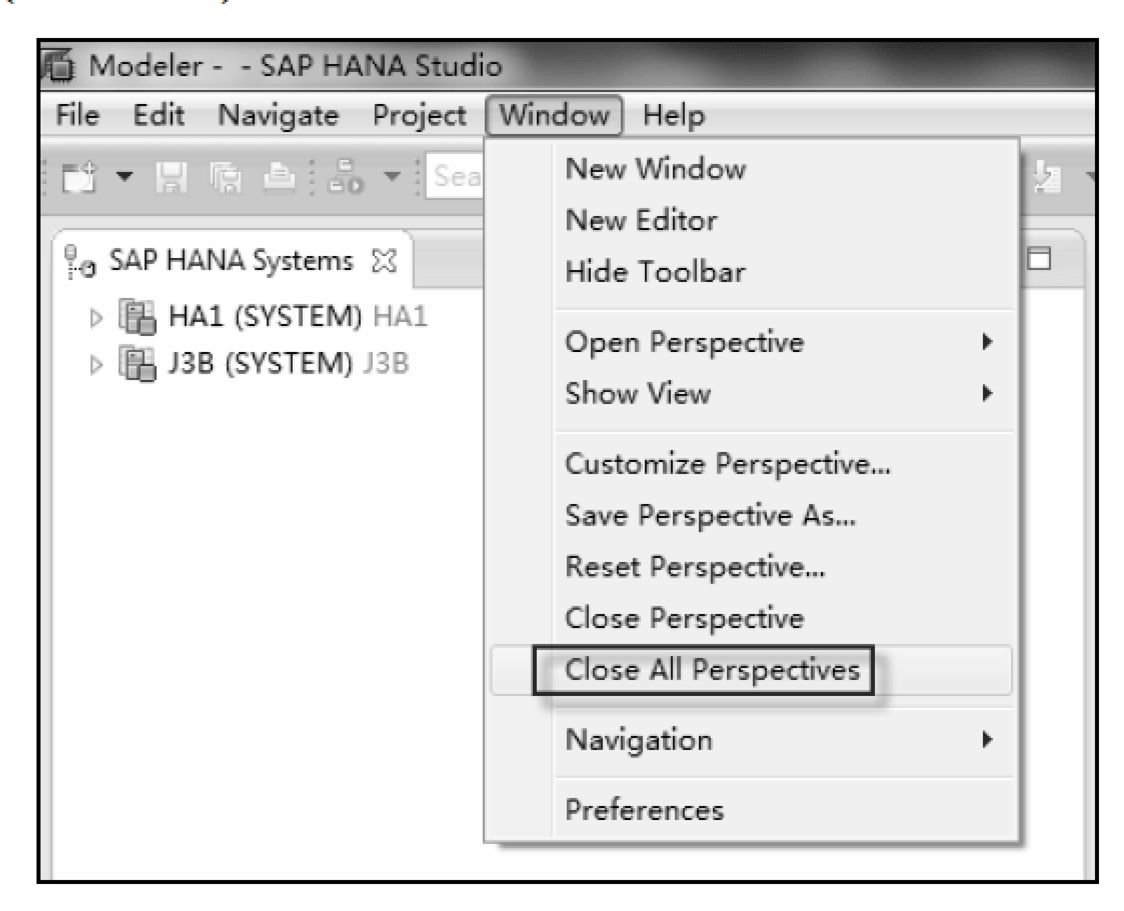


图 5-25



(2) 单击“Open Perspective”按钮(见图 5-26)。



图 5-26

(3) 在列表中选择你要定制化的透视图，我们在本例中选择“JavaScript”透视图，单击“OK”按钮(见图 5-27)。

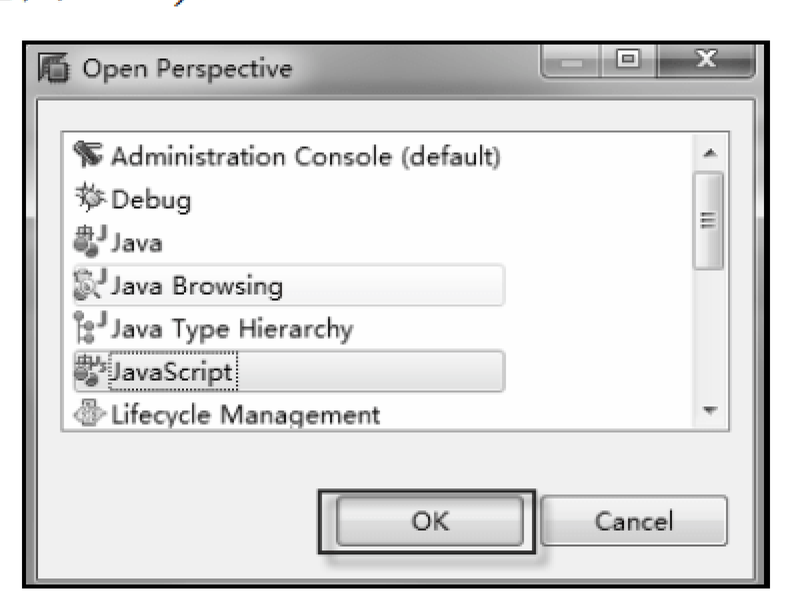


图 5-27

(4) 默认的“JavaScript”透视图组成如图 5-28 所示。

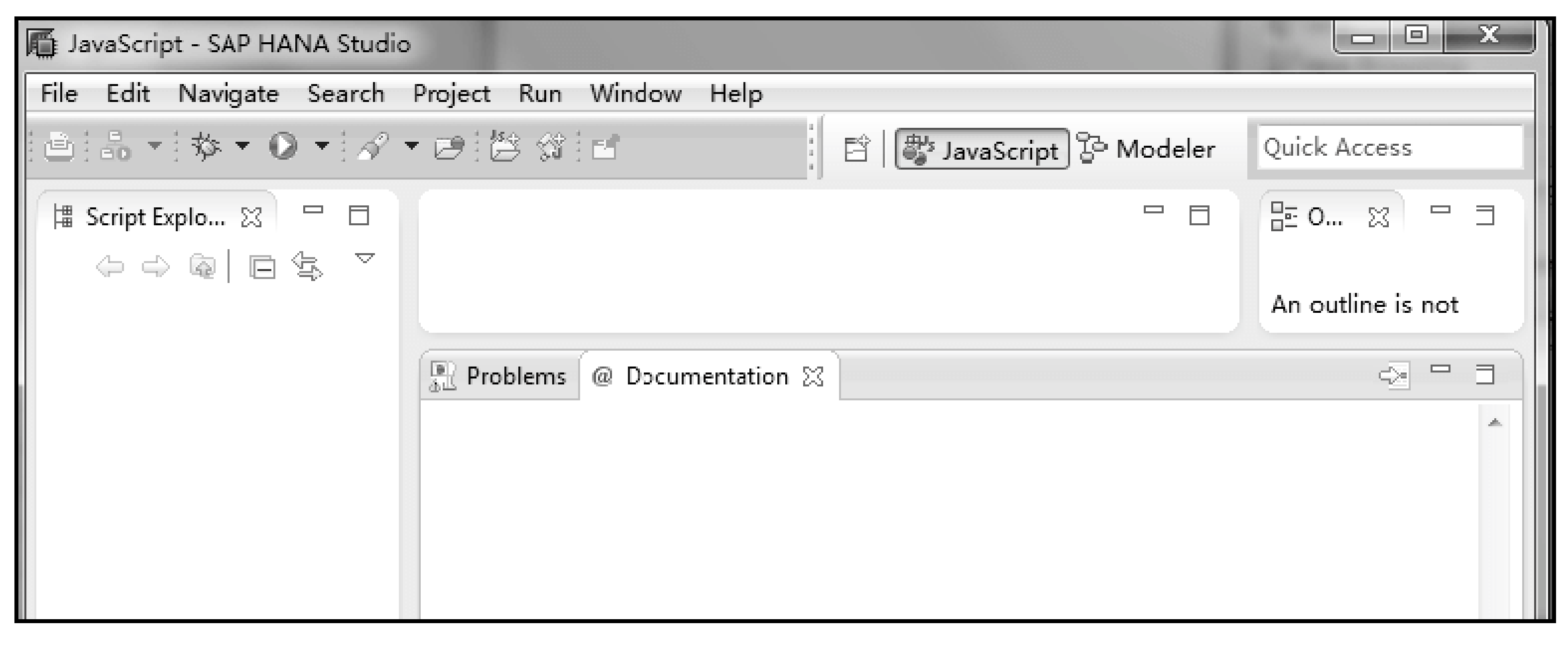
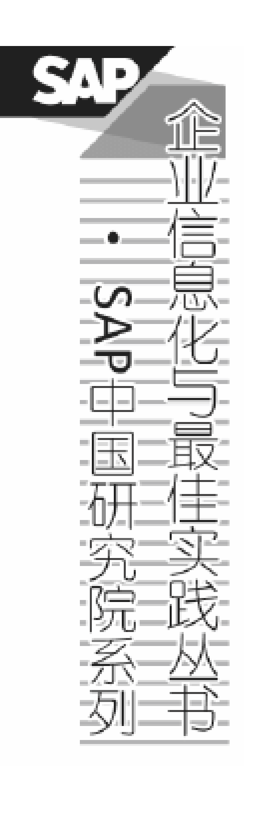


图 5-28



接下来我们为“JavaScript”透视图添加“Console”(控制台)视图来方便我们调试代码，方法为：工具栏“Window”→“Show View”→“Console”(见图 5-29)。

当然，你可以通过这个方法添加别的视图到这个透视图中。

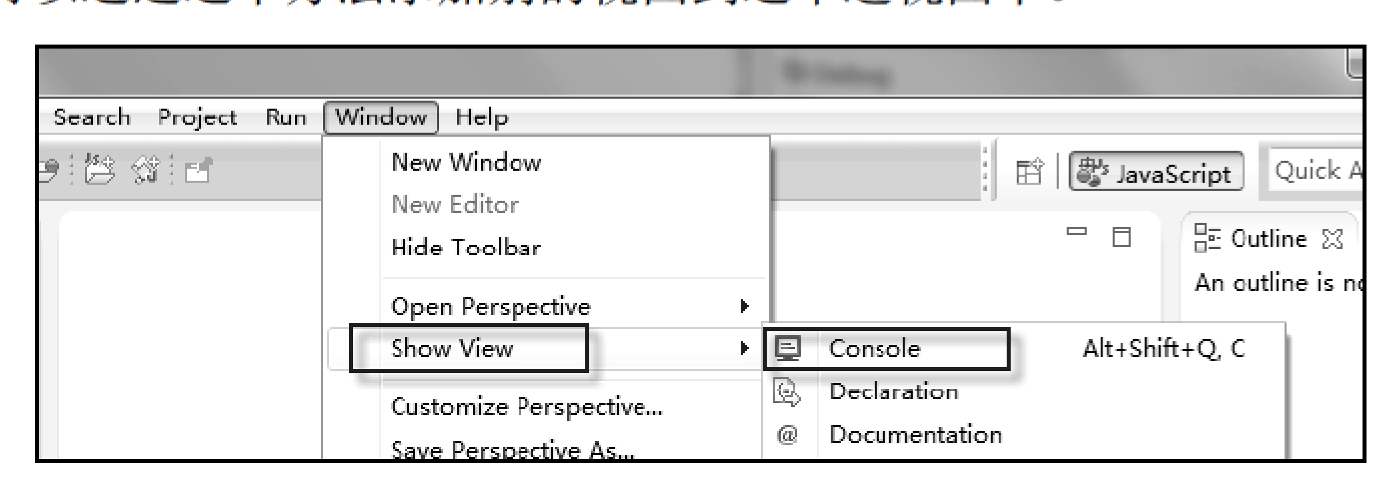


图 5-29

(5) 添加完毕的透视图如图 5-30 所示。

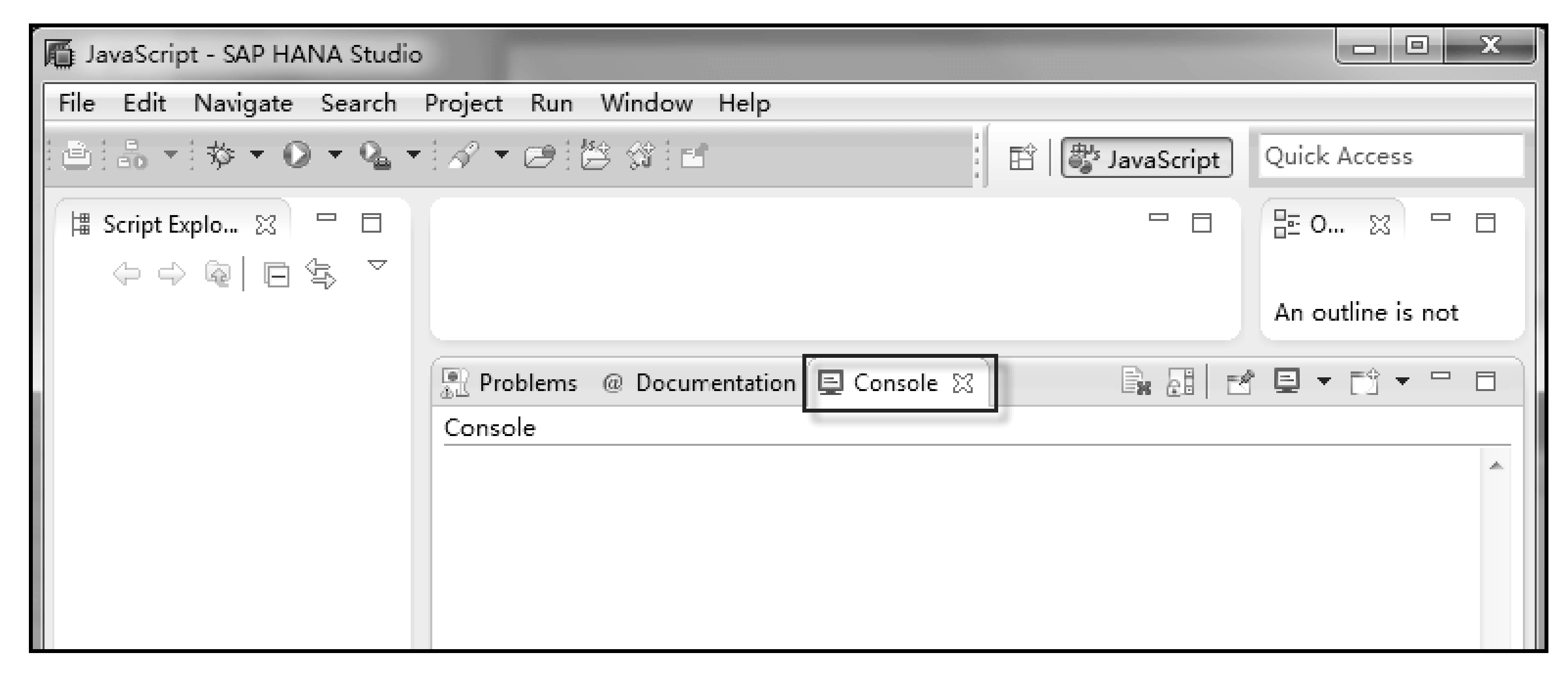


图 5-30

(6) 为了区别于系统自带的“JavaScript”透视图，我们把我们刚定制的透视图保存成自定义透视图，方法为：工具栏“Window”→“Save Perspective As...”(见图 5-31)。

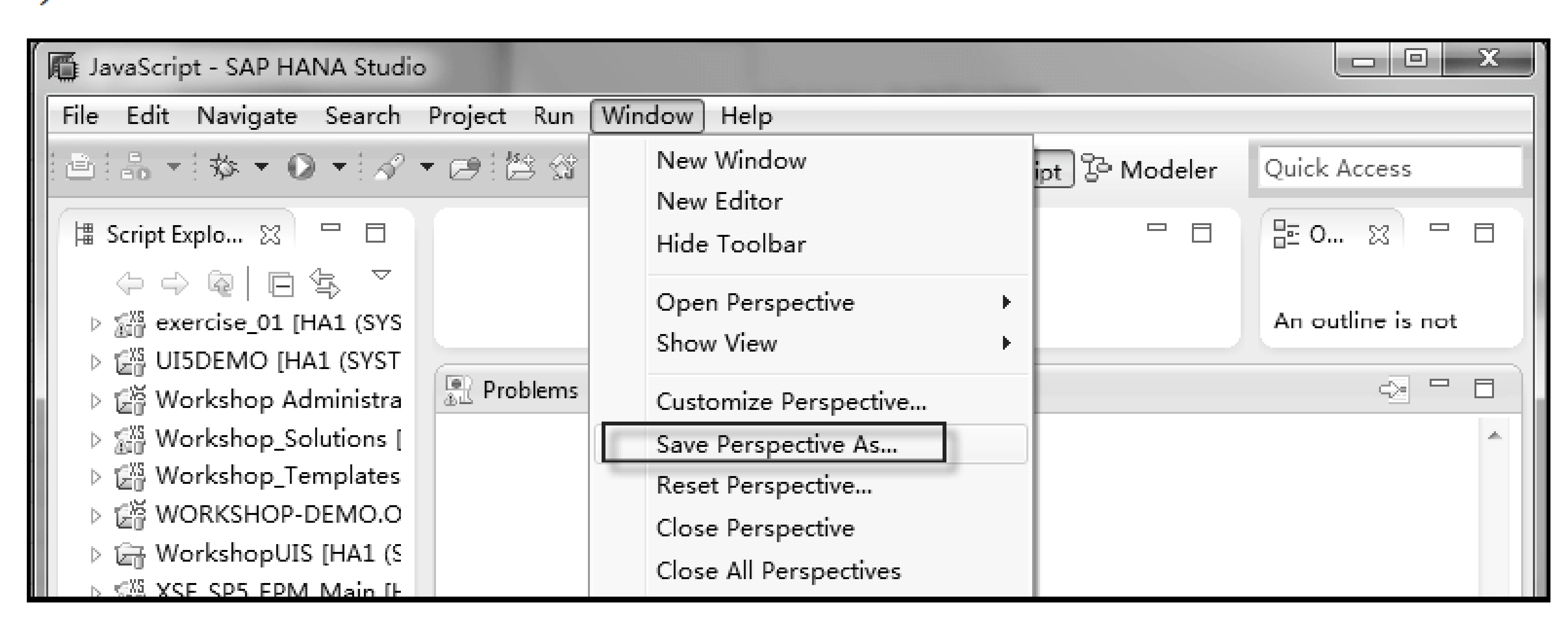


图 5-31



在弹出窗口中输入新透视图的名称，单击“OK”按钮保存(见图 5-32)。

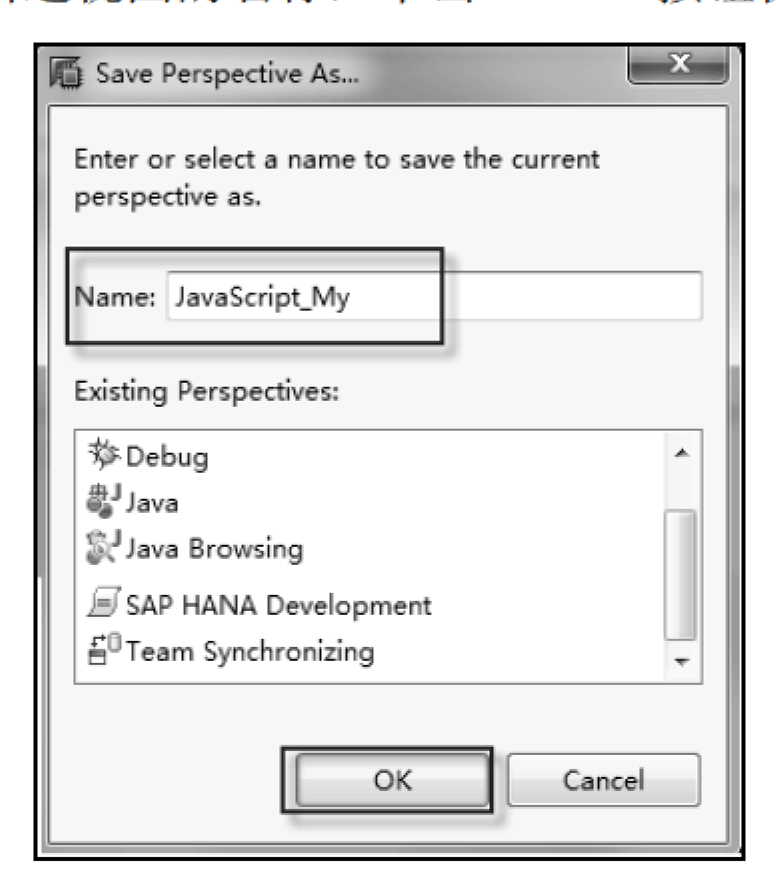


图 5-32

至此，我们完成了自定义透视图的基本操作，大家可以根据需要来在系统中自定义透视图以满足日常的开发工作。

第二节　SAP HANA Studio 视图



前面我们介绍了 SAP HANA Studio 中的透视图。在本小节中，我们将介绍一下 SAP HANA Studio 中的视图。在 SAP HANA Studio 中，所有的功能都是通过单独的视图来体现的，例如系统视图(System View)、任务视图(Task View)、搜索视图(Search View)、调试视图(Debug View)，等等。对于初学者来讲，比较常用的视图有“SAP HANA System”视图、“Quick Launch”视图、“SQL Console”视图等，下面来详细了解下这些视图。

一、系统视图

系统(SAP HANA Systems)视图以树状图的方式列出了你本机的 SAP HANA Studio 所连接的所有系统以及系统包含的各个对象(见图 5-33)。

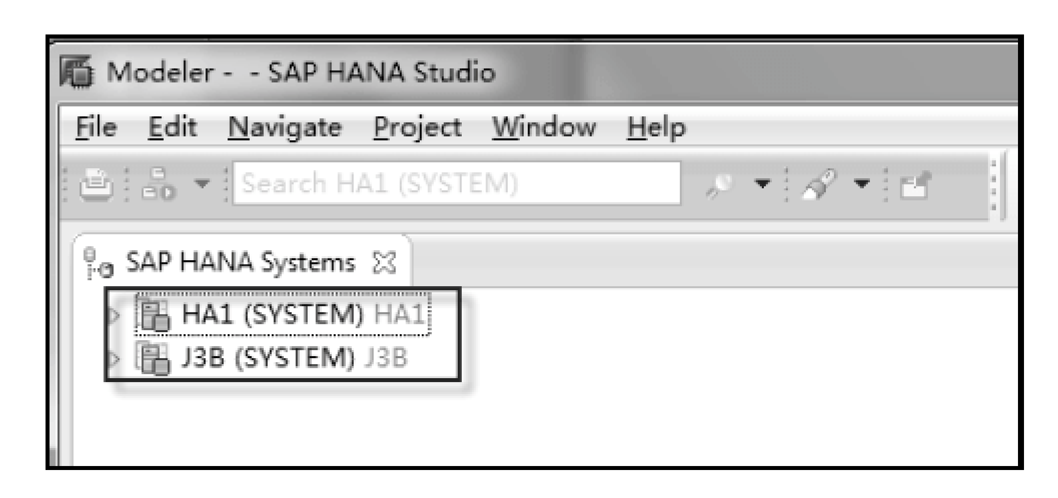


图 5-33

我们展开任一系统的节点，可以发现每个系统都包含有如下 4 个子节点，即“Catalog”、“Content”、“Provisioning”和“Security”(见图 5-34)。

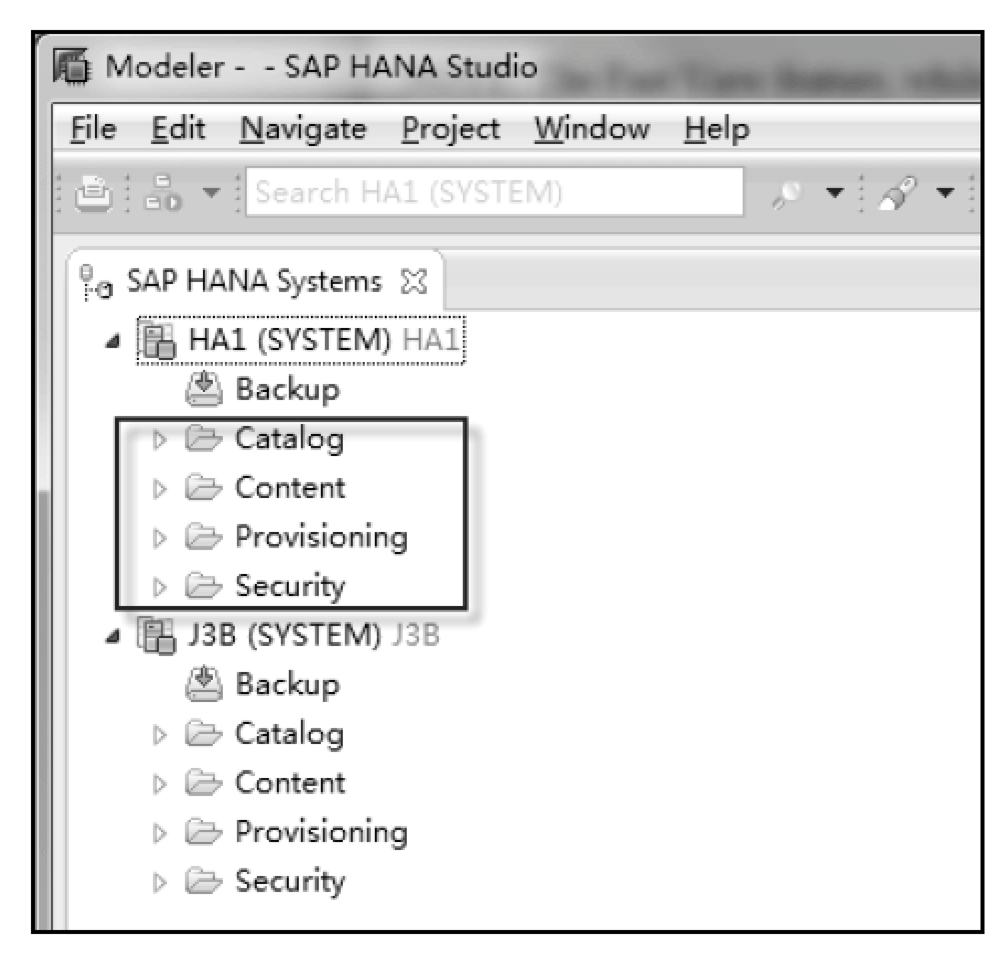


图 5-34

在“Catalog”节点中，你可以找到所连接的 SAP HANA 实例中全部激活的数据库对象(Database Object)，例如数据表(Tables)、SQL 视图(Views)、列视图(Column views)以及存储过程(Procedures)等。为了更好地为这些对象归类，我们通常将这些对象聚合到一起后放到一个 Schema 里面进行管理，并且我们可以在“Catalog”节点中创建多个 Schema 来管理不同用途的数据库对象。图 5-35 列出了我们在本书中的案例 SAP EPM 在“Catalog”节点中的 Schema(SAP_HANA_EPM_DEMO)及其所包含的数据库对象所归类的文件夹，如“Column Views”、“Tables”等。

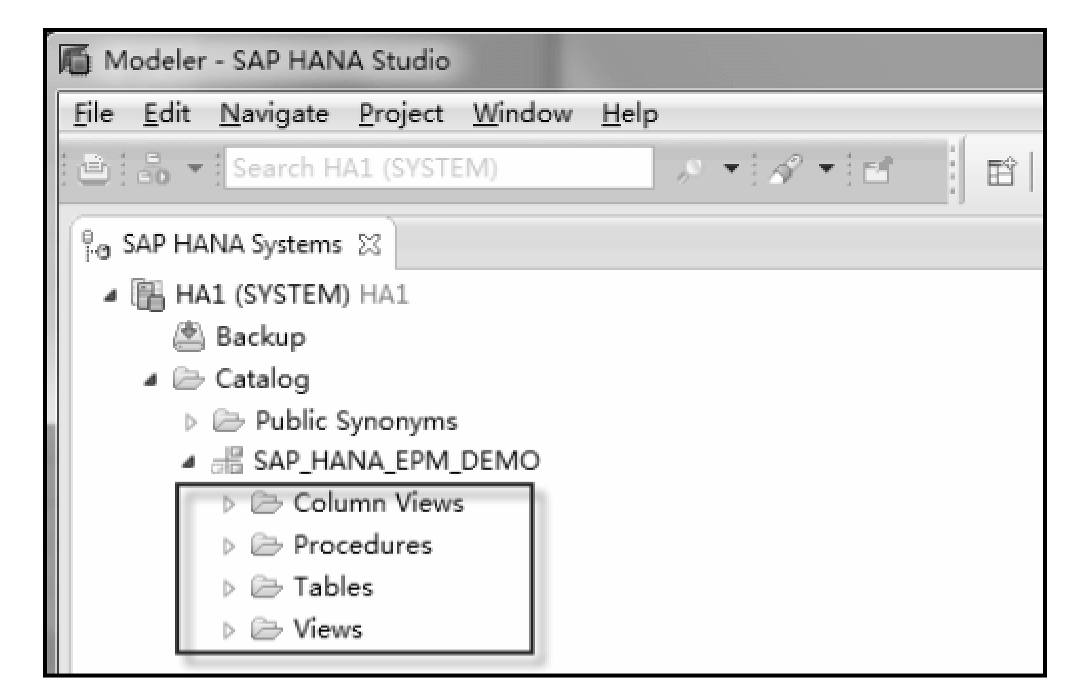


图 5-35

在以后的章节中，我们会详细介绍这些文件夹内的数据库对象以及如何创建它们。下面我们看看“Content”节点中包含哪些内容。按照如图 5-36 所示的路径，你能打开我们将要介绍的 EPM 案例在“Content”节点中的内容。仔细观察你就能发现，“Content”节点主要包含 SAP HANA Studio 的包及其子包，在包中包含诸如属性视图(Attributes Views)、分析视图(Analytic Views)等内容。同样的，我们也会在今后的章节中对这些知识点做详细的介绍。

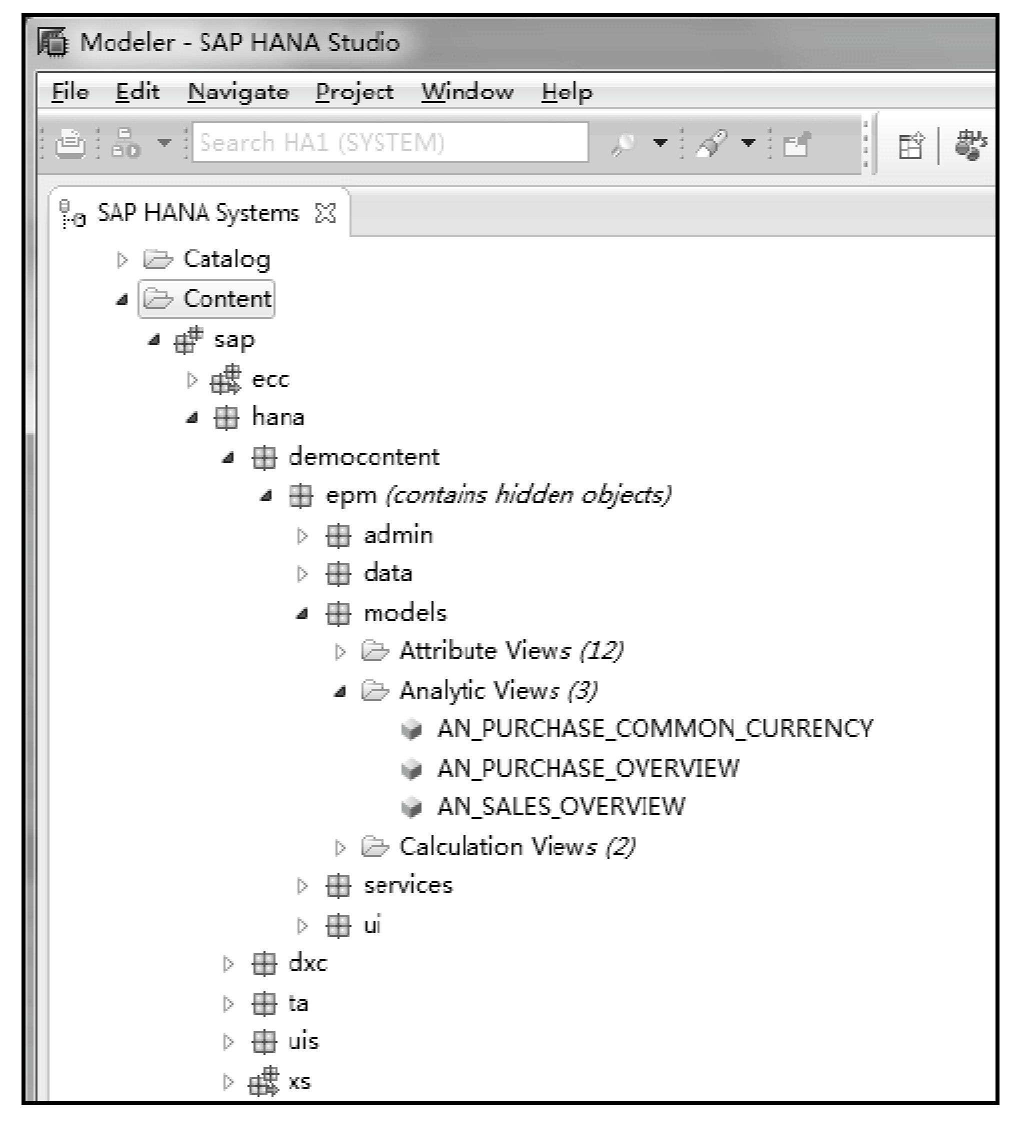


图 5-36

接下来的节点是“Provisioning”和“Security”。这两个节点的详细内容会在第六章和第七章介绍，我们就不一一赘述了。

二、快捷视图

快捷(Quick Launch)视图是 SAP HANA 初学者在 SAP HANA Studio 中最常用到的视图。我们说它最常用，是因为它提供了很多快捷方式来让开发者迅速完成开发相关的工作而不需要费力寻找相关选项的具体位置。从图 5-37 我们可以看出，快捷视图主要由 5 个区域组成，它们分别为“New”、“Content”、“Setup”、“Data”以及“Help”。接下来我们分别对这几部分做详细说明。

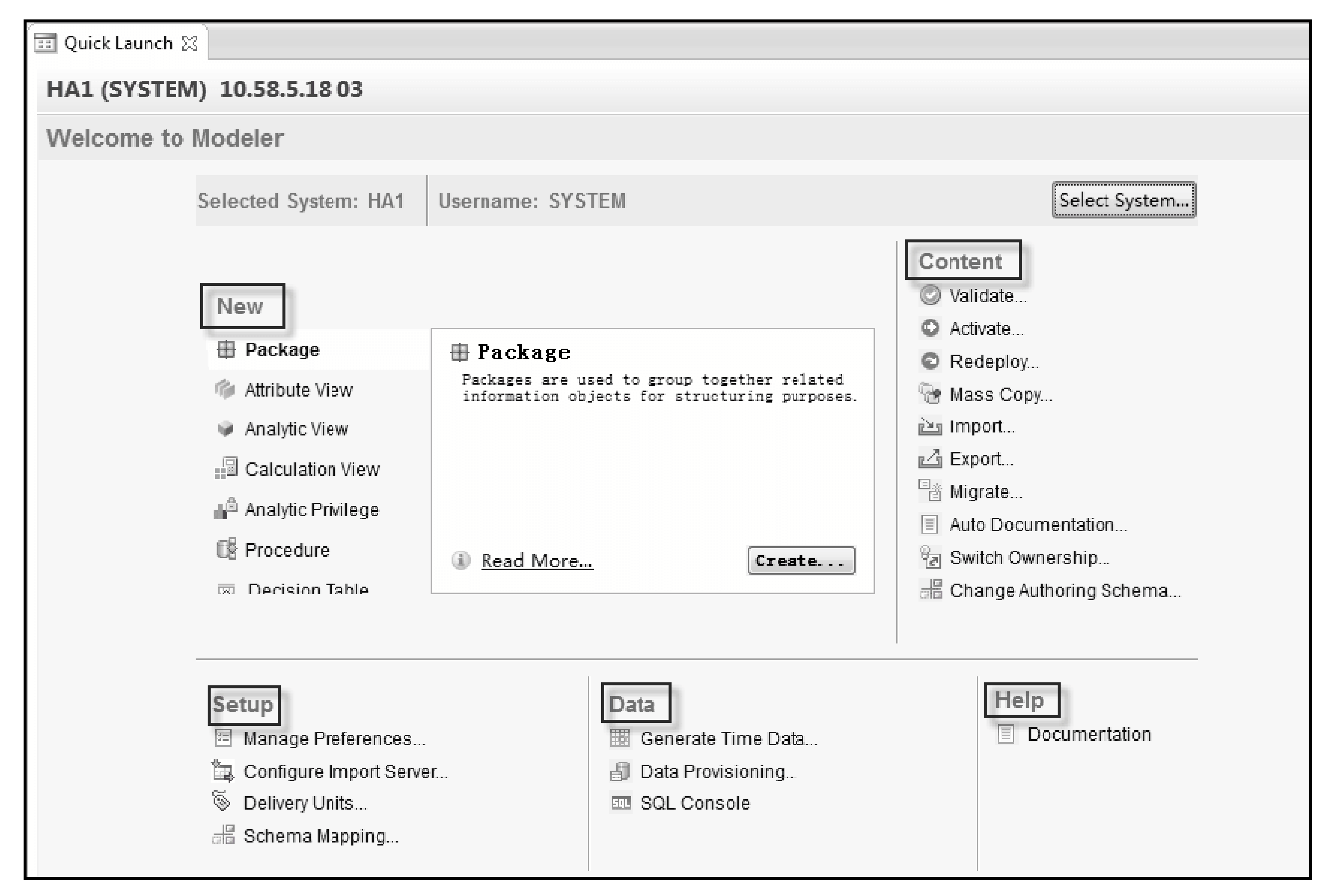
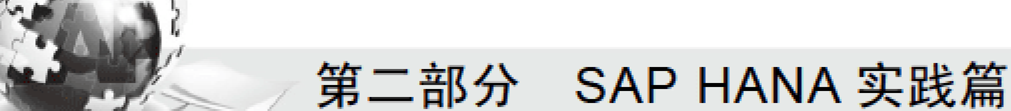


图 5-37

在对快捷视图进行操作前，你需要为快捷视图指定要关联的 SAP HANA 实例，尤其是你的 SAP HANA Studio 连接到多个 SAP HANA 实例时，这个操作尤为重要。快捷视图关联实例的操作如下：

(1) 在快捷视图上单击“Select System”按钮(见图 5-38)。





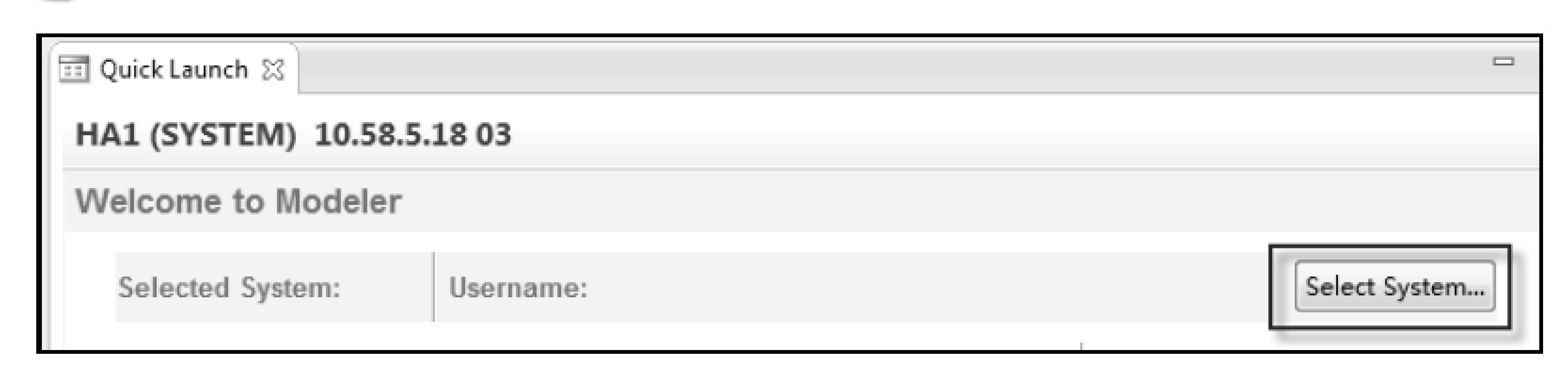


图 5-38

(2) 在弹出的系统列表中，选择你要关联的系统，例如本例中我们选择“HA1”，单击“OK”按钮确认(见图 5-39)。

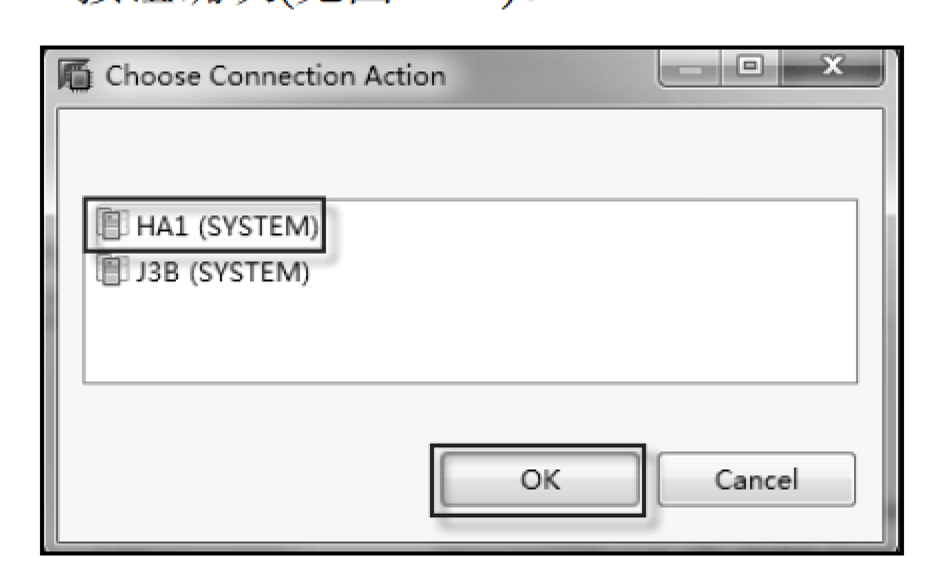


图 5-39

关联好系统后，你能在快捷视图的状态栏中看到关联系统的名称和操作用户名称(见图 5-40)。

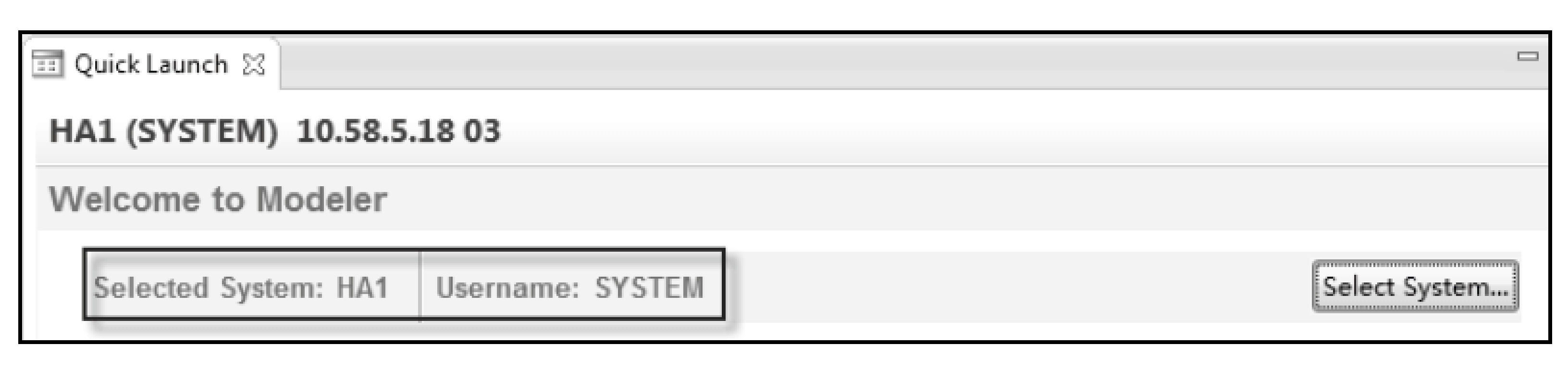


图 5-40



好了，做完准备工作后，我们逐一看看刚才提及的 5 个区域。

① **“New”区域**：如图 5-41 所示，在“New”区域中我们能完成 SAP HANA Studio 大部分的创建新对象操作，例如包(Package)、数据模型视图(View)、存储过程(Procedure)等。“New”区域分为左右两个子区域，当你在左边区域(红框内)单击需要创建的对象时，你能在右边区域(绿框内)的信息栏里面找到关于此对象的描述。单击“Create...”按钮即可打开相应对象的创建窗口。



图 5-41

② “Content”区域：如图 5-42 所示，这一区域里面提供了很多 SAP HANA Studio 内容相关的非常有用的功能，例如对于对象的验证(Validate)、激活(Activate)和重新部署(Redeploy)等功能，这些功能会在随后的章节里面进行介绍。我们先来介绍几个同样常用的功能，即“Mass Copy”、“Auto Documentation”以及“Switch Ownership”功能。

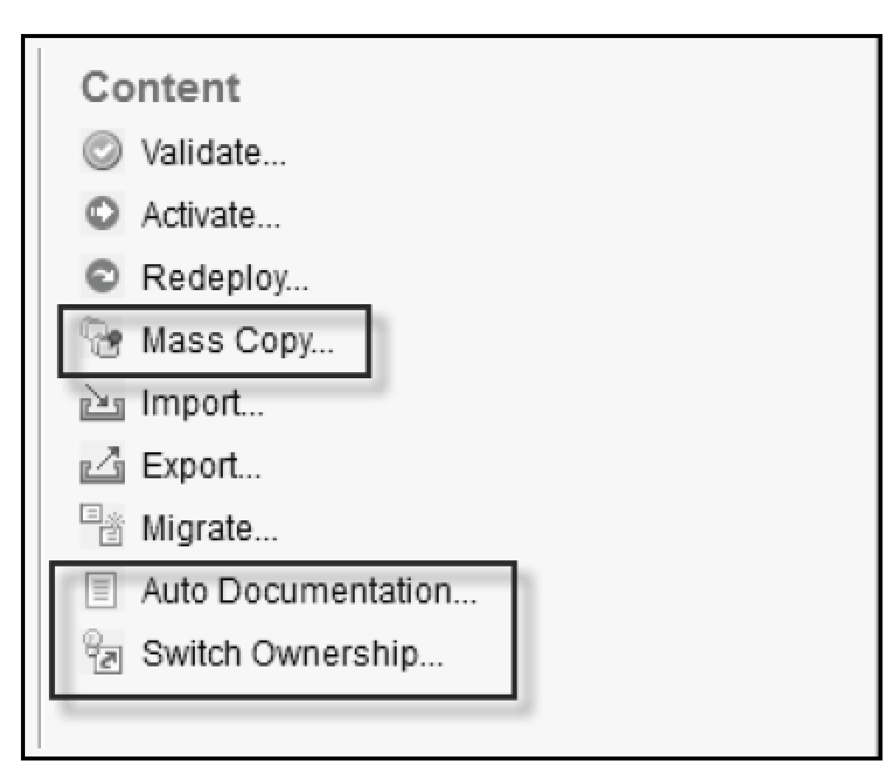
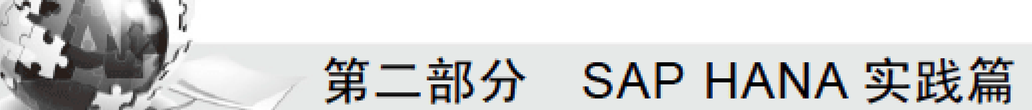


图 5-42

“Mass Copy”功能用来将一个包里面的数据模型批量复制到另外一个包里面。这个功能在复制包内容方面非常有用，这样你就不用逐个复制包内的对象了。具体操作如下，首先单击“Mass Copy”进入功能主界面，然后单击“Add”按钮，在“Source”栏中选中源包，在“Target”栏中选中目标包，单击“Next”按钮(见图 5-43)。



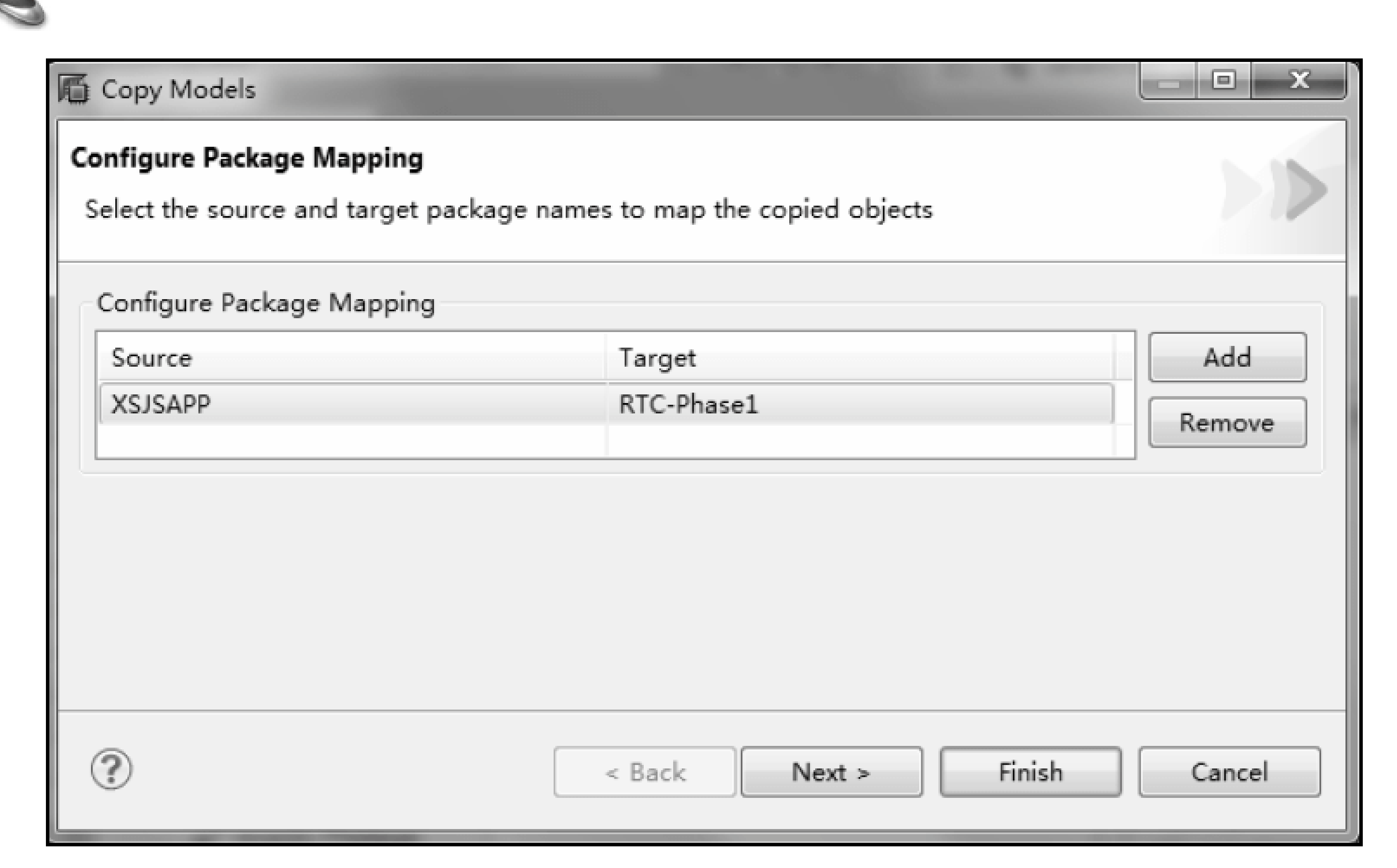


图 5-43

在下一界面里展开包，选中要复制的对象，单击“Add”按钮。你可以按住“Ctrl”键进行复制对象的多选操作。全部添加完成后，单击“Next”按钮(见图 5-44)。

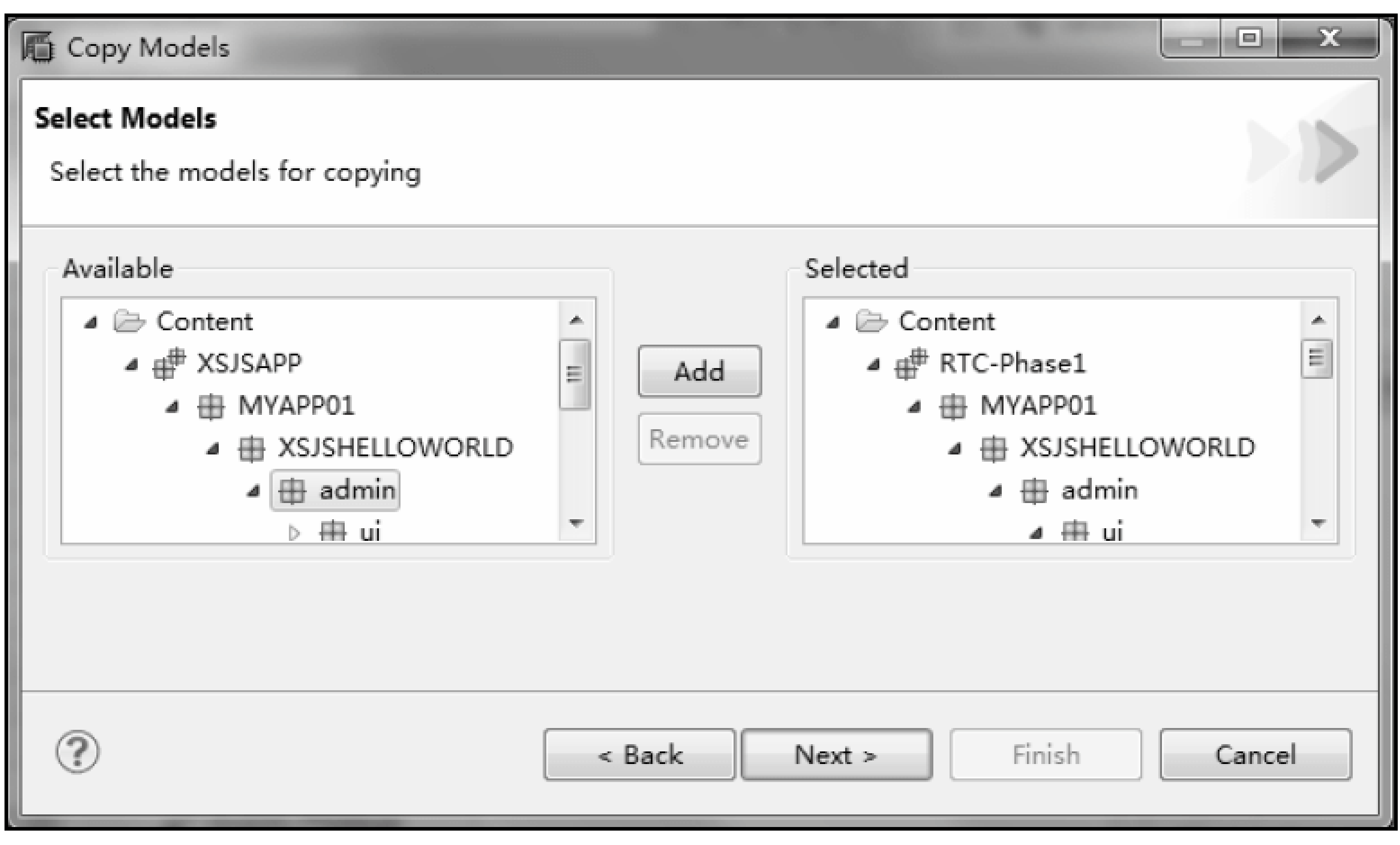


图 5-44



在接下来的界面里面你能看到复制操作的结果，单击“Finish”确认结果(见图 5-45)。

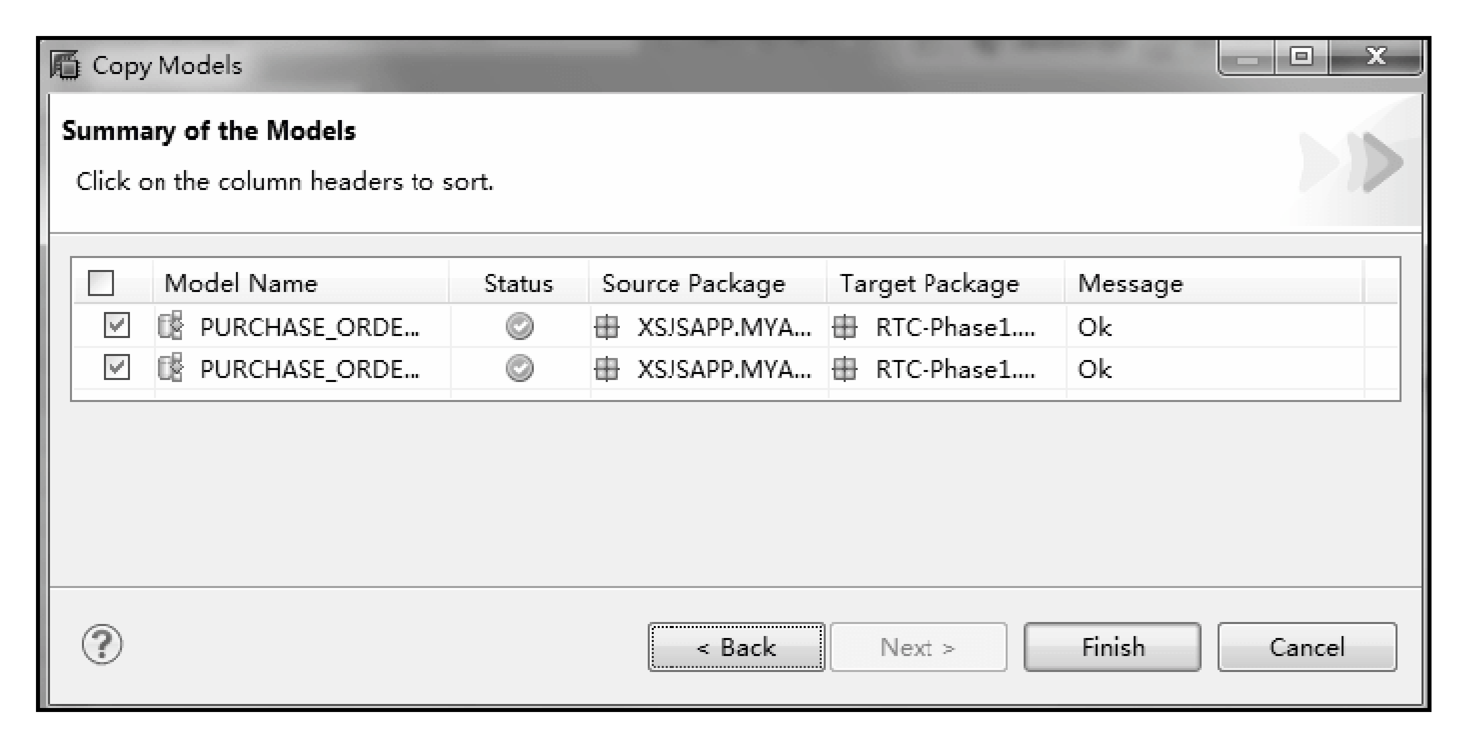


图 5-45

“Auto Documentation”功能是用来自动生成数据模型相关文档的。有了它，我们能对所创建的模型进行自动生成相关的文档的操作。具体操作为：单击“Auto Documentation”进入功能主界面，展开包找到你要生成文档的模型，按“Ctrl”键可以同时选中多个模型，选中后按“Add”按钮后，选中的模型会被传送到目标(Target)区域。单击“Browse”按钮选择文档输出的路径，最后单击“Finish”按钮形成文档生成任务(见图 5-46)。

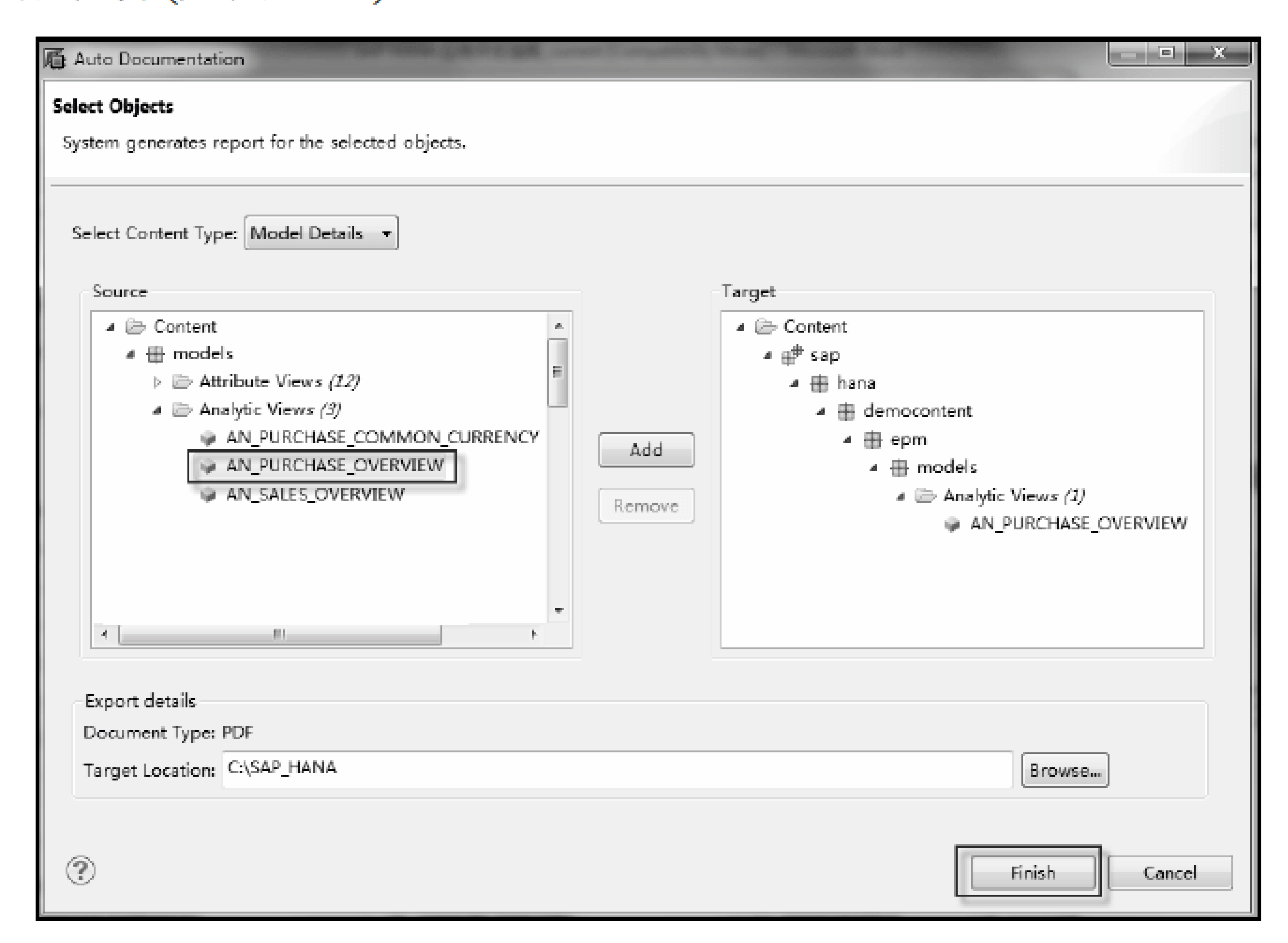


图 5-46

你会在“Job Log”视图里面找到你刚刚进行的文档生成任务，当它的状态变为“Completed successfully”时，你就能在指定的文档输出路径文件夹里面找到刚刚自动生成的文档。文档通常以“*.PDF”格式保存，你需要安装 Adobe Reader 来查看这些文档(见图 5-47)。

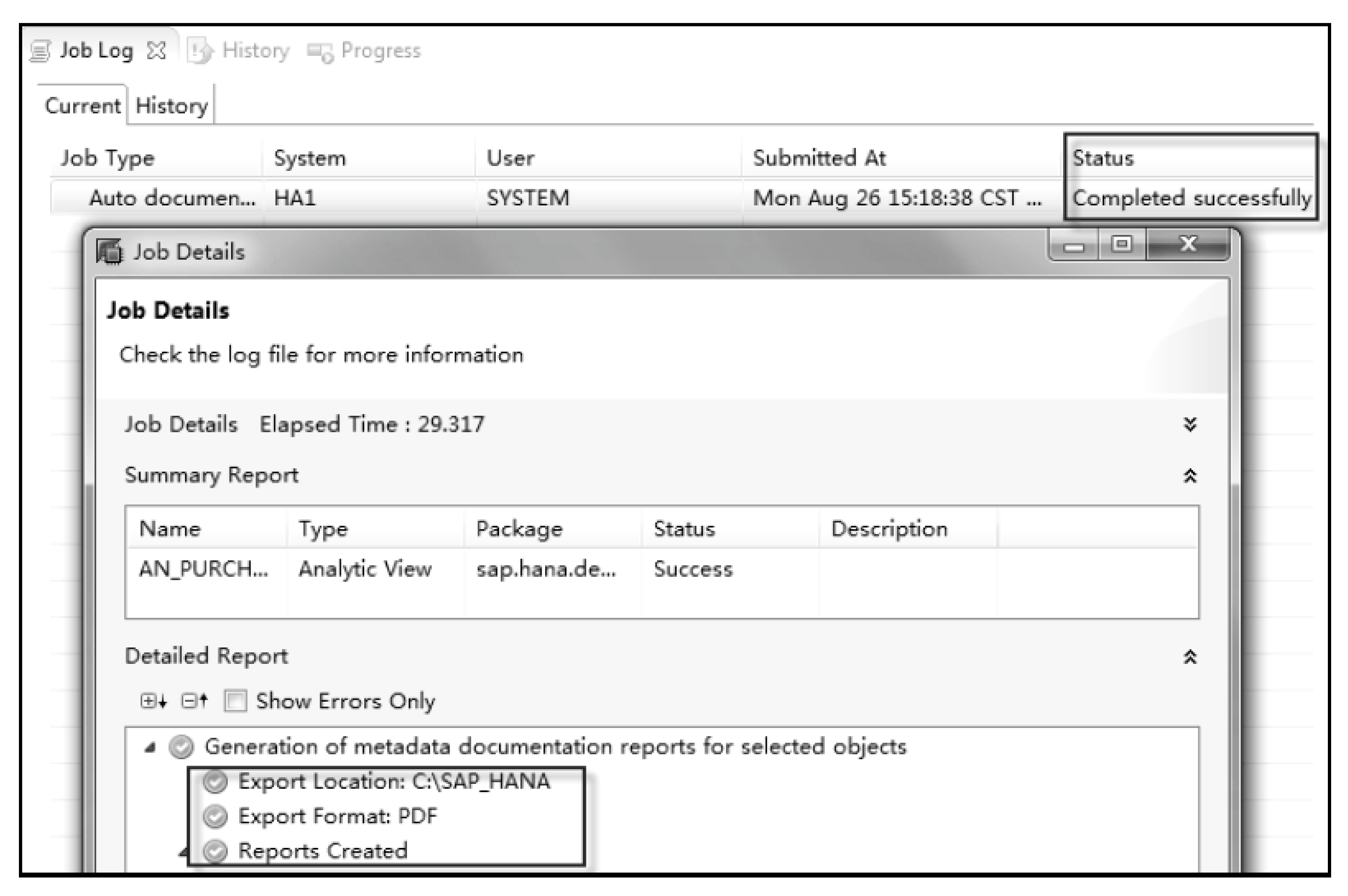


图 5-47

在你刚才指定的输出路径下找到对应的 PDF 文件，这里要注意的是，SAP HANA Studio 会在你指定的路径下默认生成系统及相关包的子文件夹，如图 5-48 所示。



图 5-48

双击文档，即可查看文档内容，如图 5-49 所示。

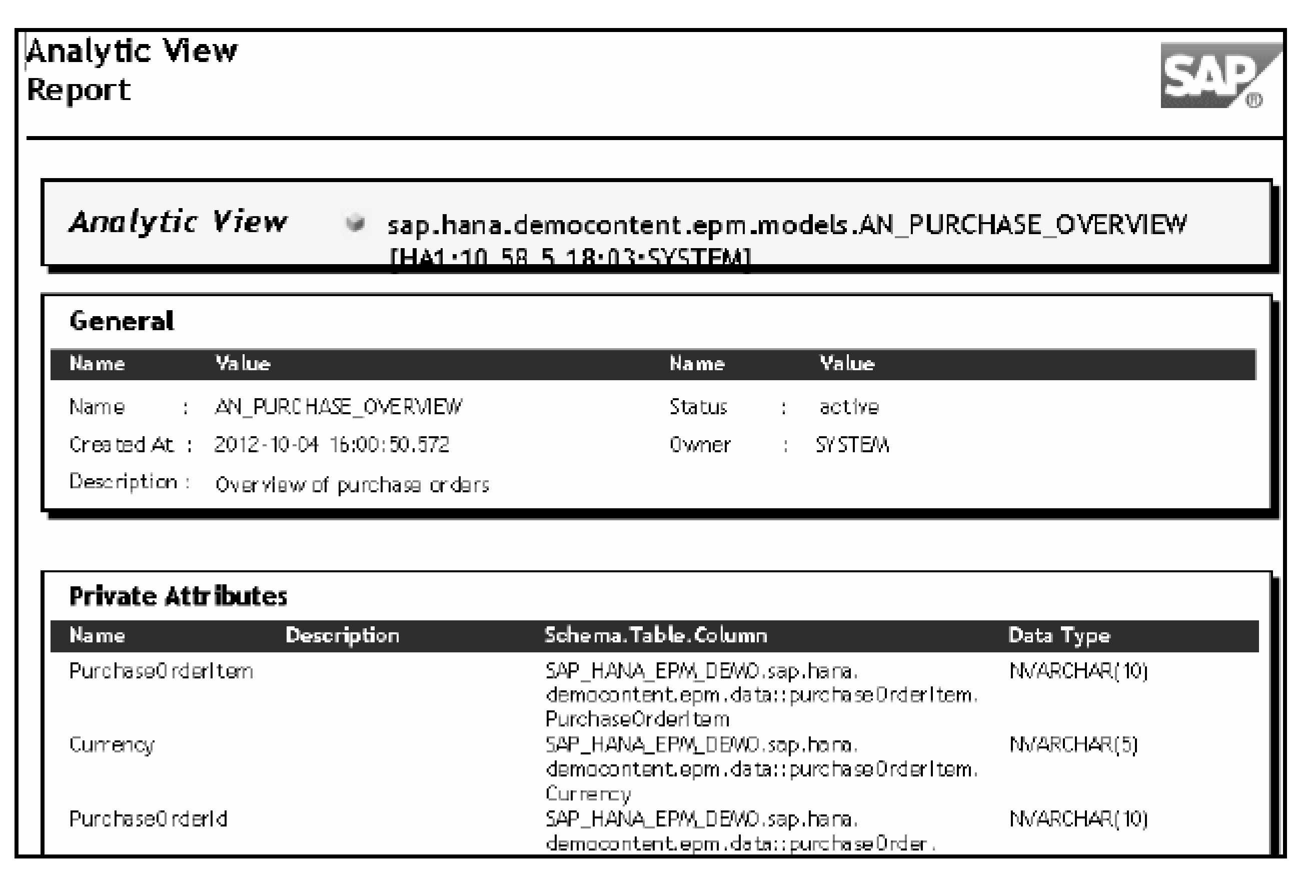


图 5-49

最后我们介绍“Switch Ownership”功能，这个功能是用来将某一用户的未激活开发对象指派给另一用户的功能。这个功能非常有用，比如你需要接替队友完成他还没做完的开发对象，或者是需要其他人帮你完成某个开发对象的工作，就会用到这个功能。具体操作为：单击“Switch Ownership”调出主程序界面，在“Source User”栏里面选择源用户，在“Model Name”列表里面选择需要分配的开发对象，你可以按住“Ctrl”键进行多选操作，将需要分配的对象全部选择完成后，单击“Add”按钮，则选中的所有开发对象会被全部转移到目标用户列表下面。单击“OK”按钮确认，如图 5-50 所示。

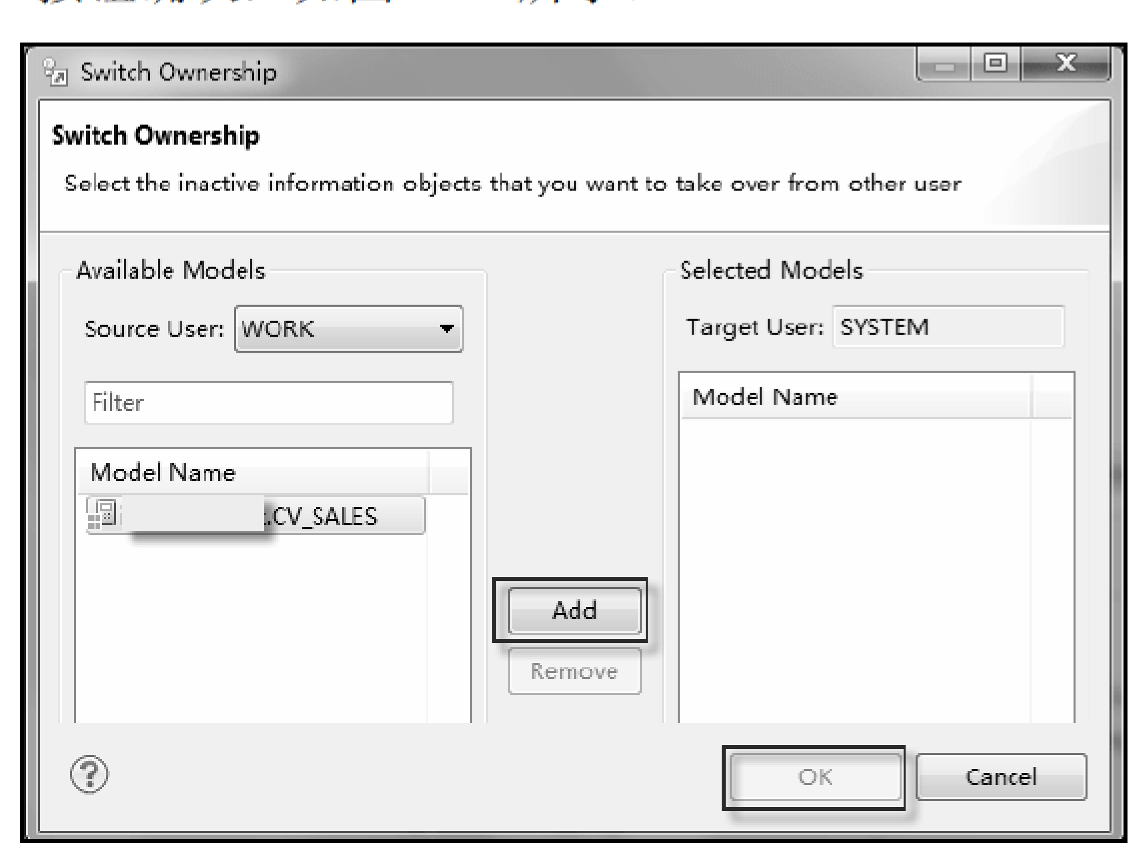


图 5-50

③ **Setup 区域**：如图 5-51 所示，“Setup”区域包含了常用的 SAP HANA Studio 配置相关的选项。

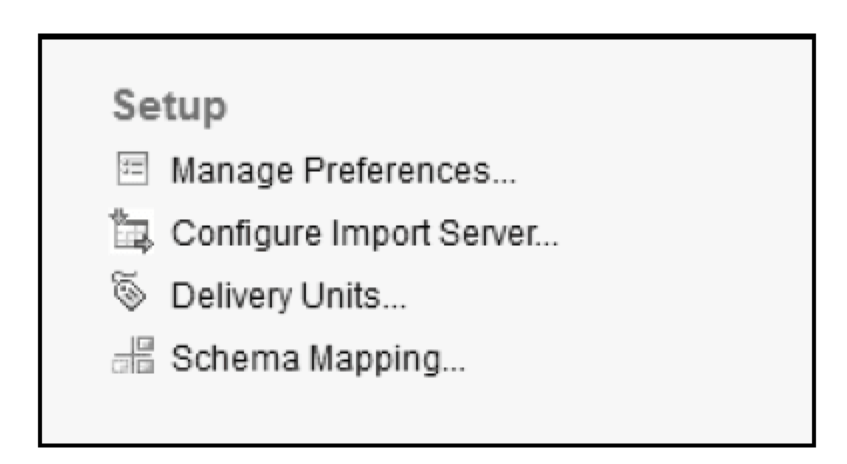


图 5-51

在这个区域里面，你能够对 SAP HANA Studio 进行参数上的配置，同时你还能定义数据输入服务器以及创建用于开发项目打包传输用的交付单元(Delivery Units)。交付单元是用于在不同的 SAP HANA 系统之间传输开发项目的对象。例如你在某个 SAP HANA 开发测试系统中进行了某个项目的开发工作后，你可以将所有的项目对象打包进交付单元内，然后传输到任一其他 SAP HANA 系统进行使用。

如果你要对 SAP HANA Studio 进行参数上的配置，你需要使用“Manage Preferences”选项，它是用来为 SAP HANA Studio 不同的透视图配置参数的地方。例如我们知道在控制台视图中，执行 SQL 语句的数据预览界面默认只提供 5000 行数据的预览操作，如果我们想预览全部数据怎么办呢？那就需要在“Manage Preferences”功能项里面为控制台视图配置数据预览的参数，具体操作为：单击“Manage Preferences”选项进入功能主界面，选中“Modeler”节点后，在右面的操作面板上单击“Data Preview”选项，如图 5-52 所示。



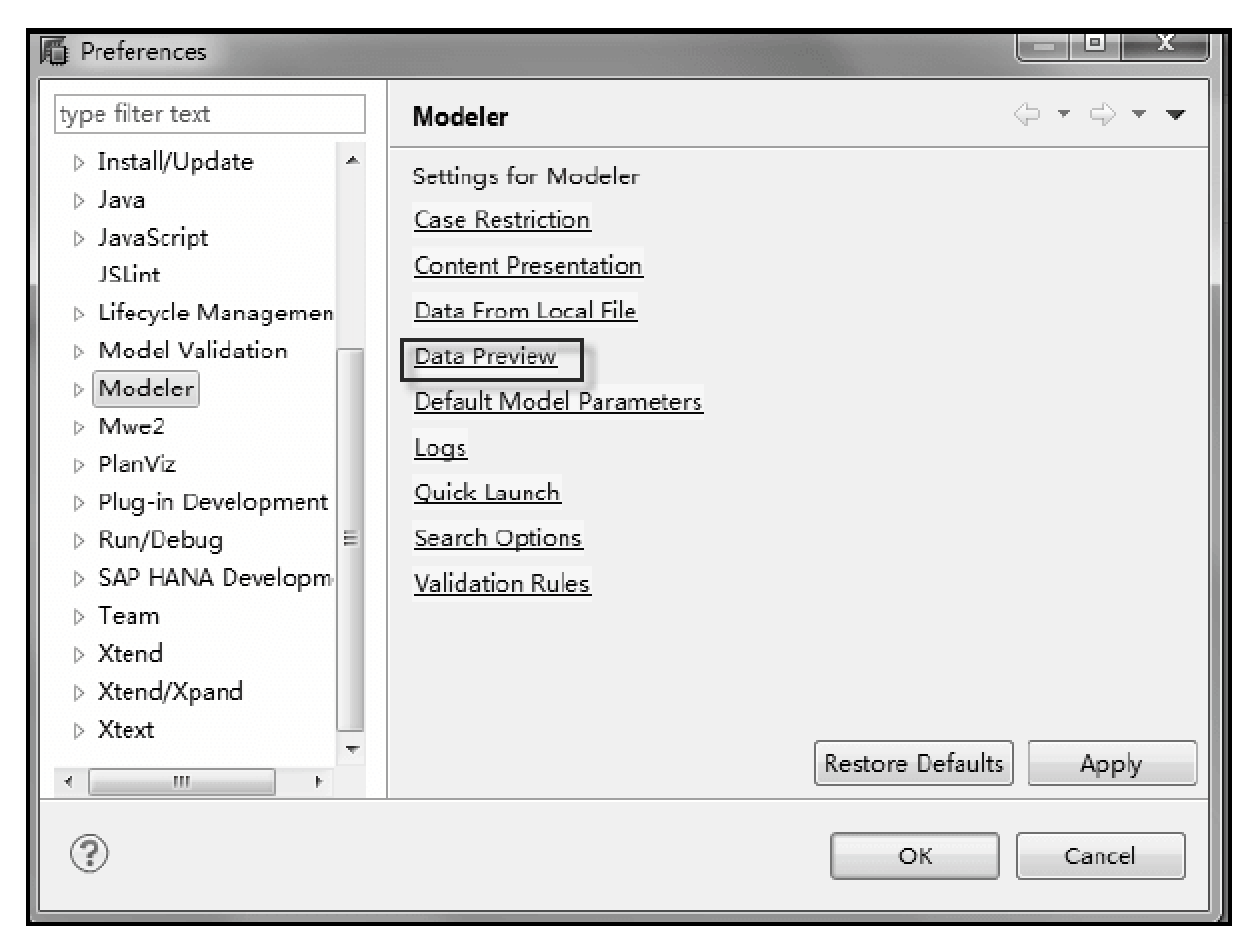


图 5-52

在随后弹出的配置界面里面选择“No Limit”→“Apply”→“OK”即可。如图 5-53 所示。

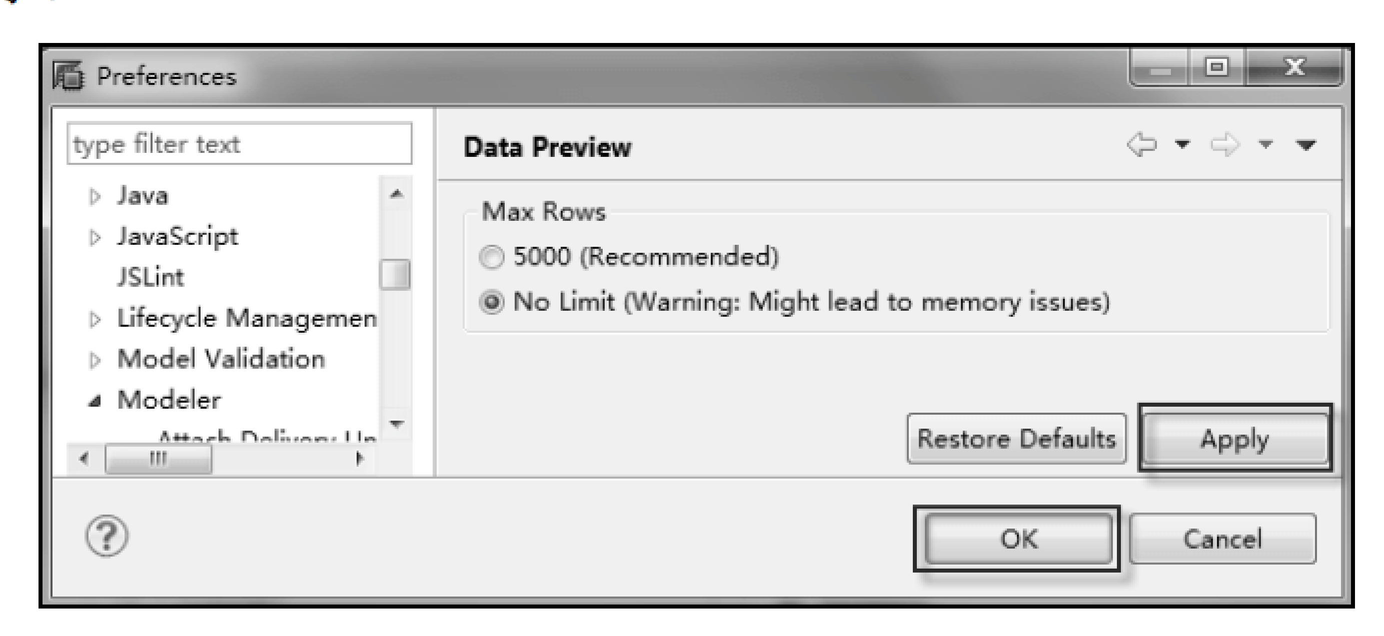


图 5-53

“Manage Preferences”功能项里面还提供很多其他的配置选项，我们在这里不一一列举，有兴趣的读者可以自己参照上面的操作尝试一下其他选项的修改，打造自己个性化的 SAP HANA Studio。下面我们说说如何配置导入 SAP HANA 外部数据的服务器参数。

SAP HANA Studio 提供了非常方便的与 SAP Data Service 集成接口来快速导入外部数据，你首先要做的就是在“Configure Import Server”选项配置外部数据服务器的参数。图 5-54 为配置“Configure Import Server”的界面，首先输入需要导入数据的外部服务器的地址，然后输入 ODBC 数据源及其他必要信息即可，有关这方面的知识我们会在第七章做详细介绍。

图 5-54



“Delivery Units”，即交付单元选项，是 SAP HANA Studio 的一个重要的特性，我们在这里先不展开解释，在随后介绍应用开发的章节里面，我们会根据具体的例子来教大家怎样使用交付单元来把你开发好的应用打包并进行系统间的传输。

④ **Data 区域**：在这个区域里面，我们可以维护数据供应(Data Provisioning)的设置，生成时间数据和打开 SQL 编辑器的工作。这些具体的功能将在随后的章节里面进行详细介绍。

⑤ **Help 区域**：这个区域提供所有的关于 SAP HANA 的帮助文档集合的链接(见图 5-55)。

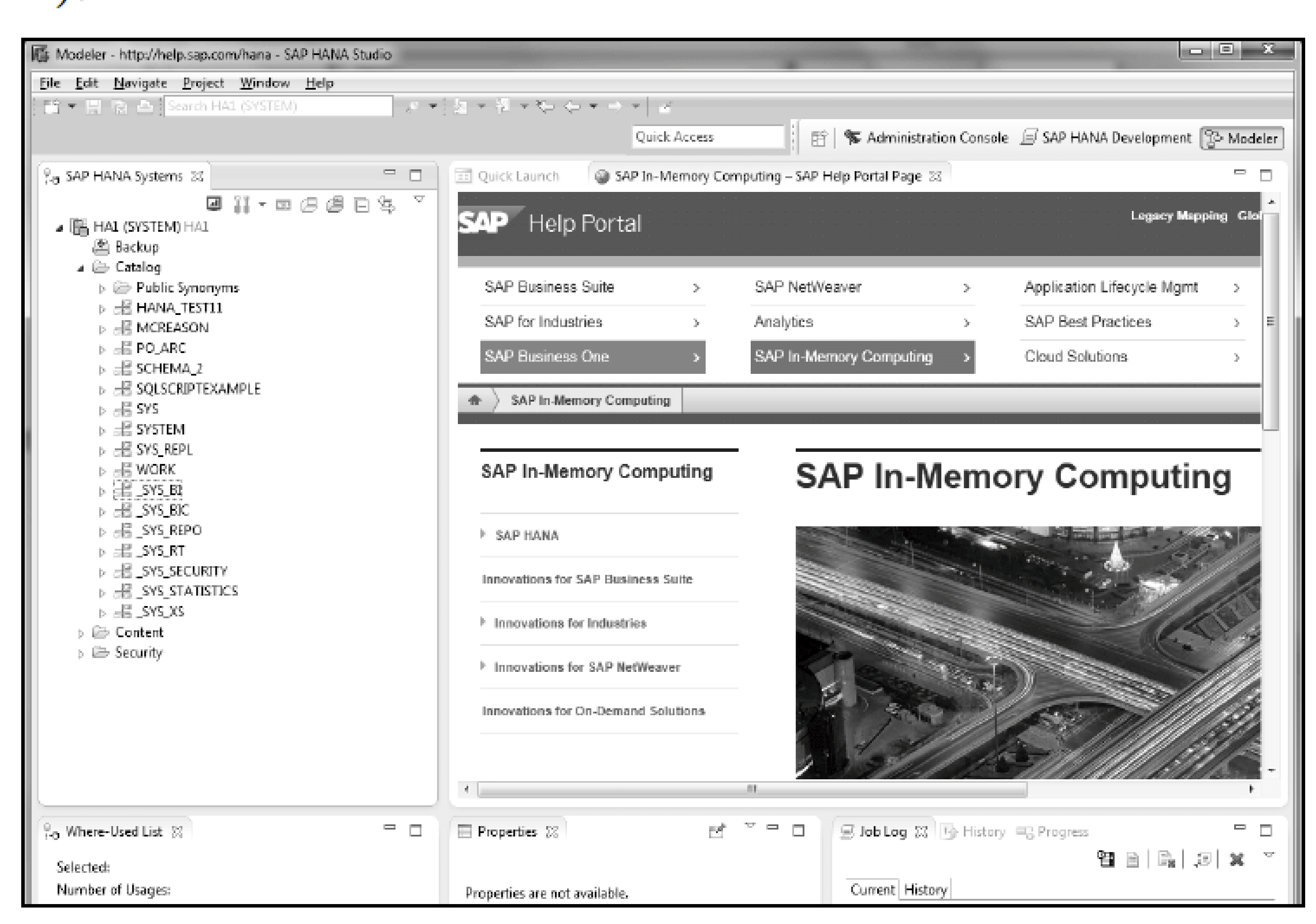


图 5-55

三、导入 EPM 交付单元

在介绍完快捷视图后，我们就能开始导入本书中的实例“SAP EPM”交付单元到系统中的工作了。

虽然在 SAP HANA SPS06 版本中，EPM 实例是默认自带的练习实例，但是除非你手工将其导入系统，否则此交付单元是不会被默认安装的，请遵循以下步骤将其导入：

(1) 打开“Quick Launch”视图，单击“Import”选项(见图 5-56)。



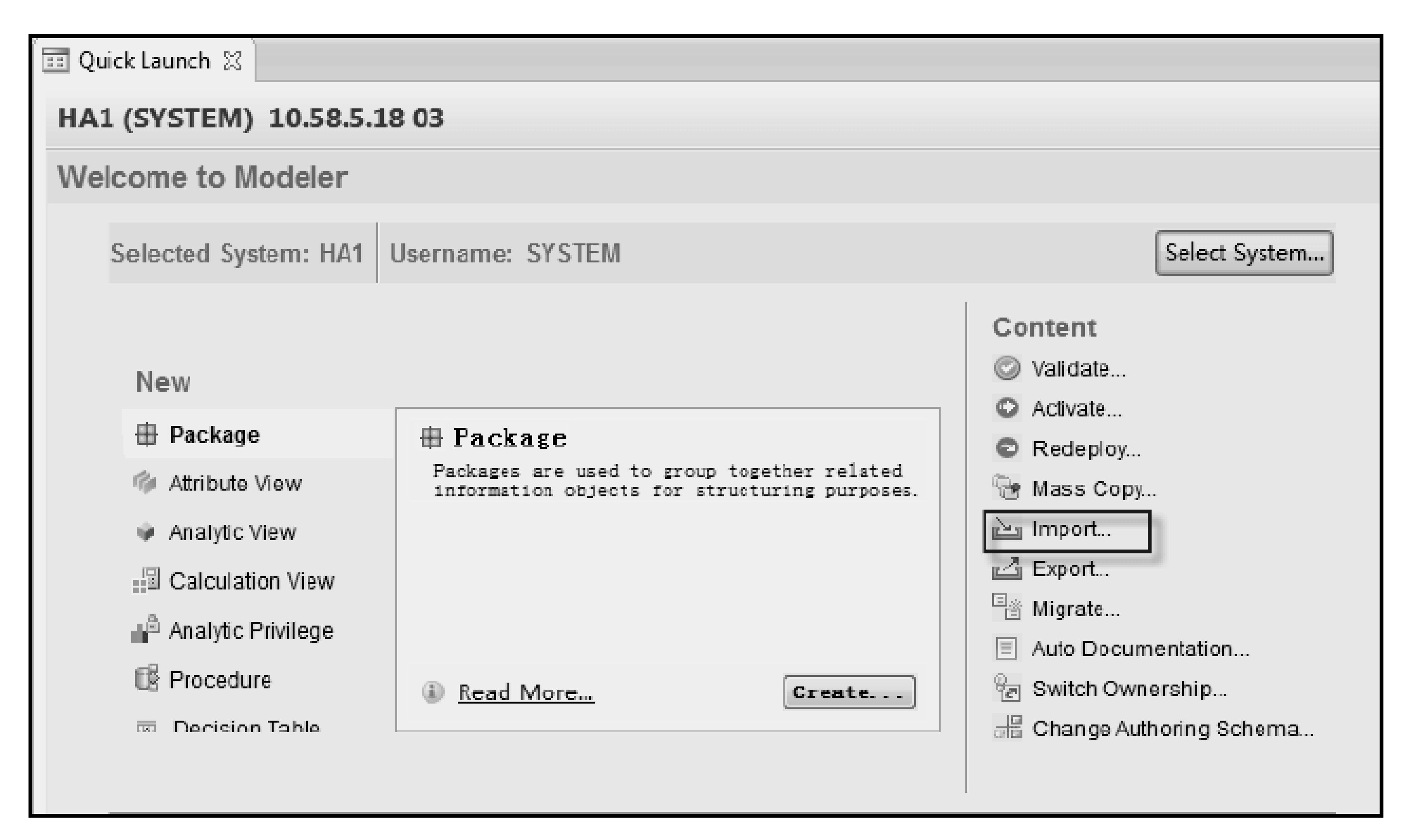


图 5-56

(2) 在弹出窗口展开“SAP HANA Content”节点，选择“Delivery Unit”选项，单击“Next”按钮继续(见图 5-57)。

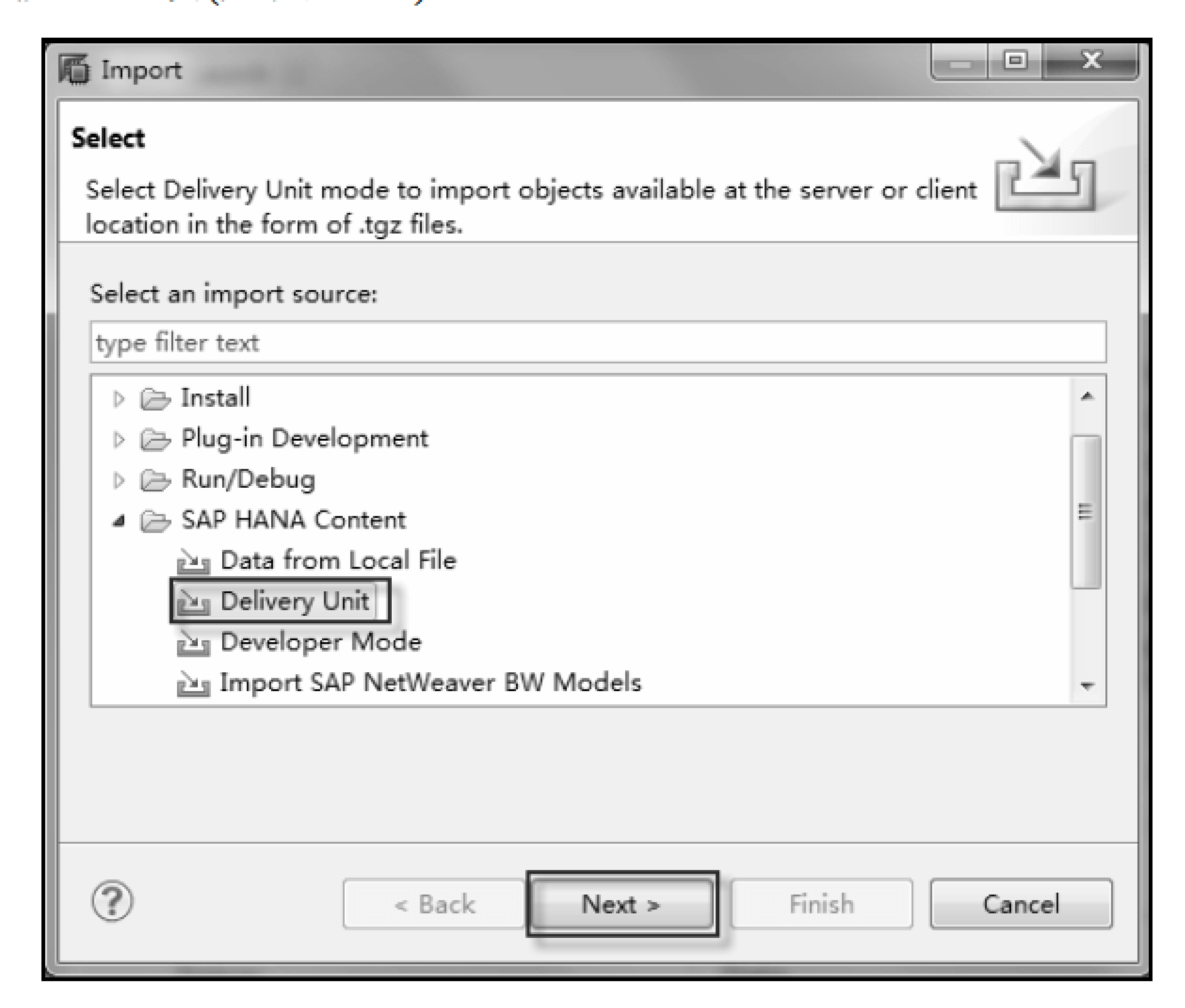


图 5-57

(3) 选择从“Server”导入(选中“Server”单选按钮)，在“File”下拉菜单中找到交付单元名称为“HCO_DEMOCONTENT.tgz”的文件，单击“Finish”按钮确认(见图 5-58)。



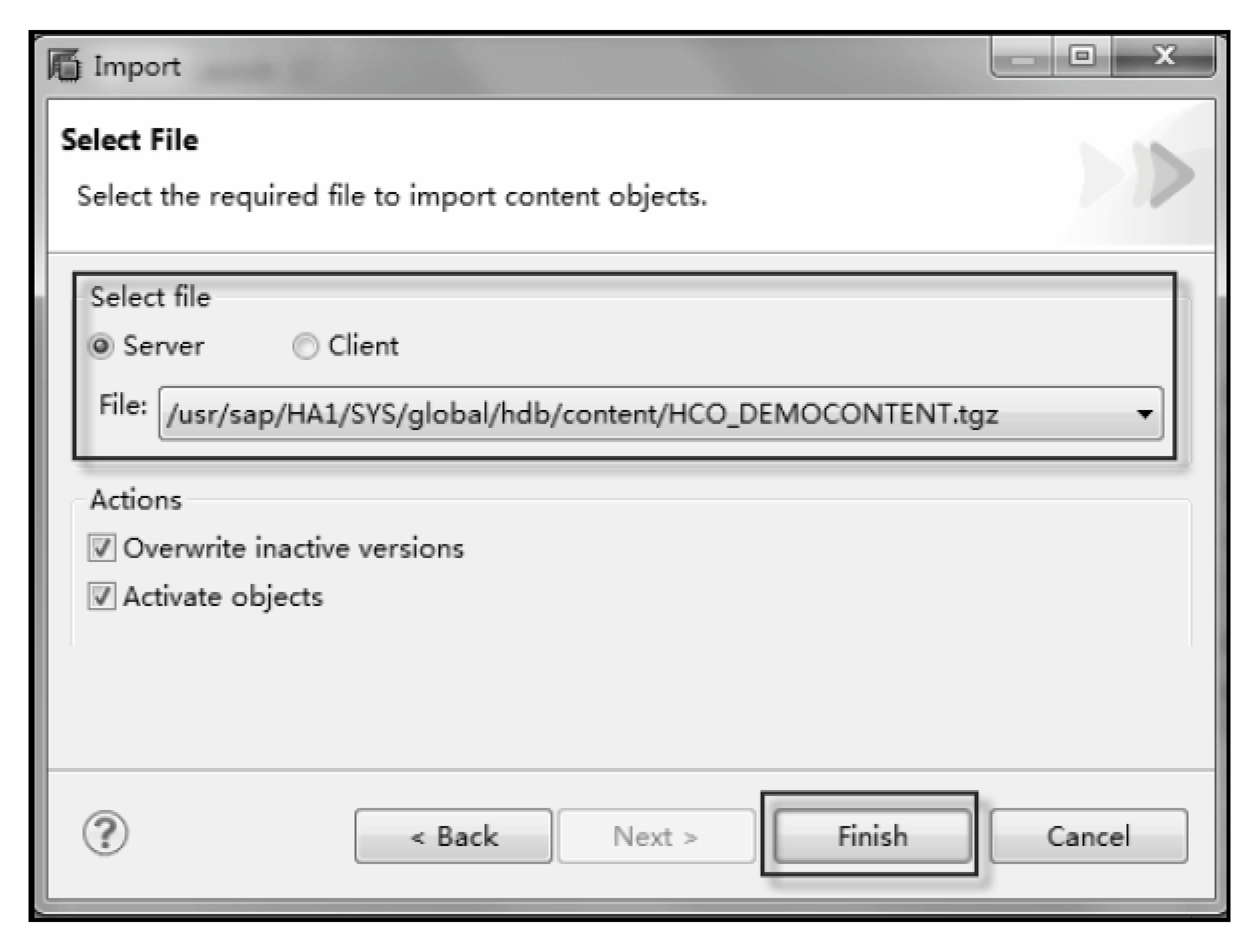


图 5-58

接下来 SAP HANA Studio 就会将 EPM 实例的全部内容导入 SAP HANA 服务器，你可以在以后章节的学习中具体来了解这些内容所处的位置和如何使用它们来进行练习。

本章小结与练习

本章的内容非常简单，我们先从如何申请 SAP HANA 实例开始，到简要介绍 SAP HANA Studio 的一些常见透视图和视图，最后我们完成了本书最关键的练习实例的交付单元的导入，接下来做些习题巩固一下吧。

练习

1. 请根据本章内容申请你自己的基于云的 SAP HANA 开发者实例。
2. 请尝试安装最新版本的 SAP HANA Client 和 SAP HANA Studio。
3. 请在 SAP HANA Studio 中尝试配置各个透视图的参数。
4. 请打开“Modeler”透视图，任意添加某一视图后保存为“Modeler_HANA”透视图。
5. 请将 SAP EPM 实例导入你的开发实例。

第六章　SAP HANA 用户与权限管理

第一节　概　述

在正式进入我们的 SAP HANA 之旅之前，我们还需要了解一下 SAP HANA 的权限与认证管理(Authorization and Authentication)。与传统数据库不同的是，由于 HANA 对硬件性能要求相对较高，在大多数的情况下，我们会与其他用户一起共享使用同一个 HANA 实例。在这个时候，如何了解并合理分配相应用户的权限以确保用户之间相互影响冲突的可能性降到最低，或是在用户发生权限缺失时找到原因迅速解决，都是一项常见又相当重要的任务。

与传统数据库类似，SAP HANA 权限可以分为三层架构：特权(Privilege)，角色(Role)与用户(User)。它们每一层之间都是多对多的关系(如图 6-1 所示)。特权是最细颗粒度的权限概念，任何操作都需要拥有对应的特权才能进行，没有特权就意味着不能进行操作。角色可以看做是“特权”的组合，我们可以在系统中定义一个角色将一些常见任务所需要的特权包含起来；再将该角色赋予需要它的用户，这样可以大大提高系统管理员用户管理的效率，但另一方面，特权可以不赋予角色而直接赋予用户。

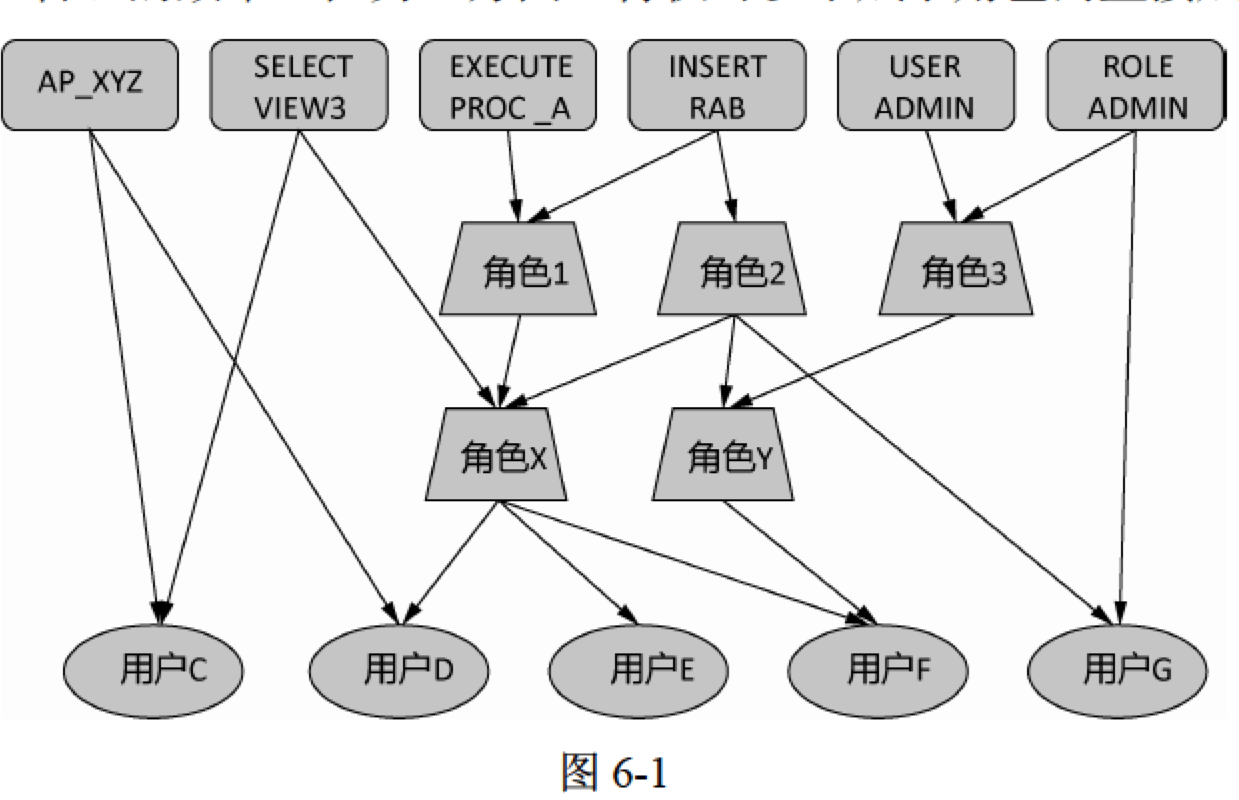


图 6-1

第二节　SAP HANA 特权

什么是特权？在 SAP HANA 中，特权是最基本的授权单元。它可以被赋予用户或者角色，也可以在赋予后又被回收。没有对应的特权则表示没有访问权限。同时，在 SAP HANA 中权限是不允许“拒绝访问”的，也就是说如果你被赋予了很多特权，只要有一个特权允许你做相应动作，就不必再检查其他特权而可以直接确定用户可以执行操作。对于特权有如下一些特点：

- 特权可以直接被赋予用户或者通过角色赋予用户。
- 推荐通过角色来对用户进行特权管理。角色是可以引用角色的，这样的好处是我们可以非常灵活地创建各种颗粒度的特权管理。
- 不允许显式去除特权。这意味着系统不需要检查用户所拥有的所有特权。一旦需要的特权被检测到，则系统自动放弃其余检查，直接认为其拥有特权。
- 有一些预定义的角色已经存在于系统中，可以被直接使用也可以作为模板来定制化。

在 SAP HANA 中，总共有以下几种主要类型的特权：

- 系统特权(System Privilege)
  - 用于管理任务，如创建用户，系统备份等
- 对象特权(Object Privilege)
  - 是否允许执行 SQL 对象相关操作
- 分析特权(Analytic Privilege)
  - 用户是否允许访问分析视图等，并对数据进行行级别限制
- Schema 特权(Schema Privilege)
  - 是否允许对 Schema 进行相关操作
- 程序包特权(Package Privilege)
  - 限制用户对分析包的访问
- 应用特权(Application Privilege)
  - 是否可以访问 SAP HANA XS 应用
  - 可通过_SYS_REPO schema 中的 GRANT_APPLICATION_PRIVILEGE 和 REVOKE_APPLICATION_PRIVILEGE 进行赋予与取消

打开 SAP HANA Studio，在如图 6-2 所示的用户或角色管理界面中，我们可以在不同的标签页中添加对应的特权，而对每一种类型的特权，SAP HANA 都预定义了很多基本值，我们将在下一节简单介绍。

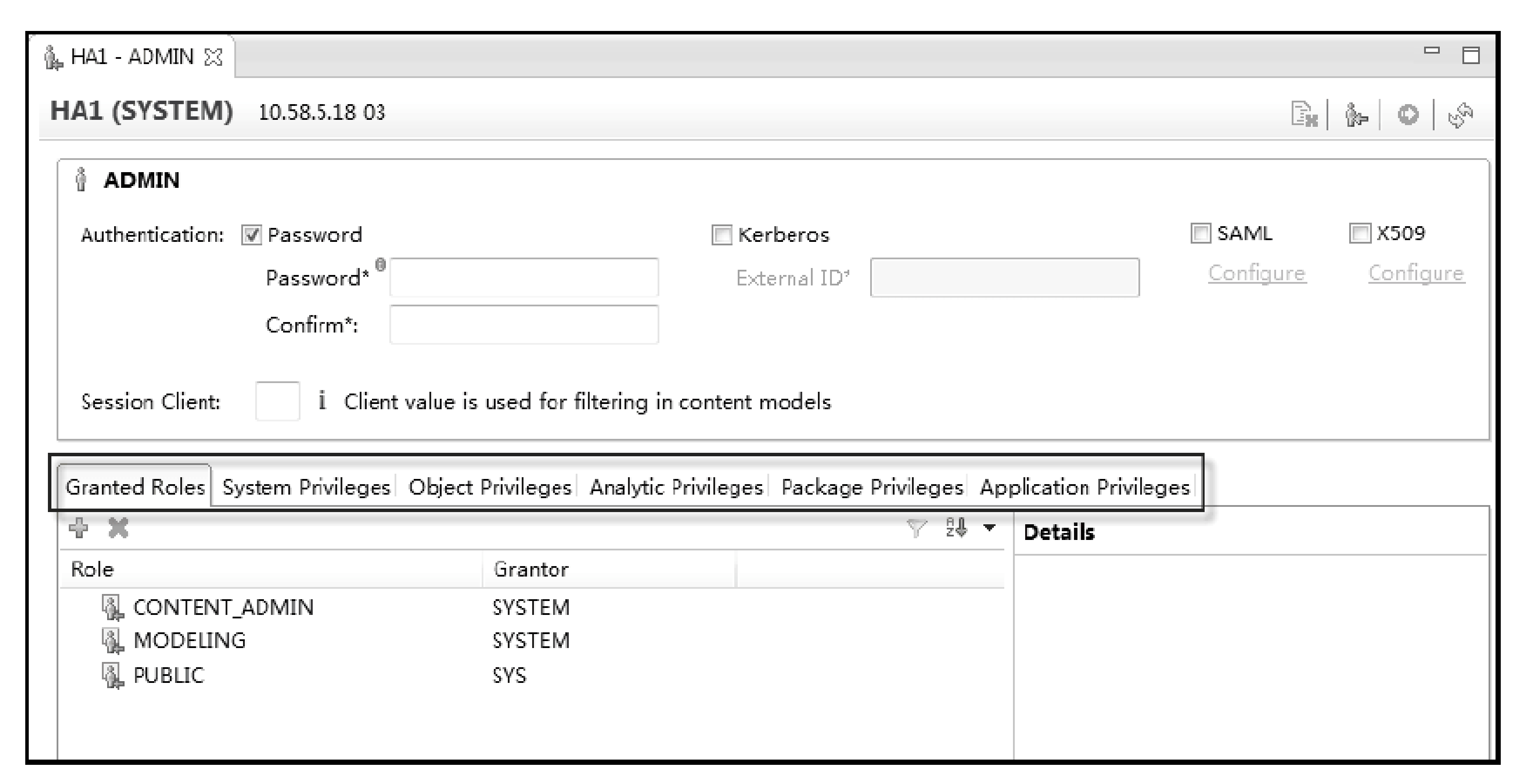


图 6-2

一、系统特权

在 SAP HANA 中，系统特权用于 HANA 系统层面的操作或管理任务。常见的系统特权包括：

- 数据库 Schema：创建与删除数据库 Schema，如 CREATE SCHEMA，DATA ADMIN。
- 用户/角色：维护管理用户角色，如 USER ADMIN、ROLE ADMIN 等。
- 监控/追踪：管理监控追踪系统运行状况，如 MONITOR ADMIN 等。
- 备份/恢复：备份与恢复操作，如 BACKUP ADMIN、LOG ADMIN 等。

在 SAP HANA 系统中 SYSTEM 用户已经拥有了所有的系统特权。在客户实际生产系统中，SYSTEM 用户不应该被使用，而是针对不同的系统任务，通过 SYSTEM 用户来创建相应的用户并赋予相应的特权来实现。

二、对象特权

在 SAP HANA 中，对象特权可以被理解成一种 SQL 特权，它决定是否能对数



据库对象进行访问或者修改。对任一对象的任一 SQL 操作类型(如 SELECT，UPDATE 或者 CALL)，都有对应的对象特权。Schema 特权是一种特殊的对象特权。当用户拥有某一 Schema 特权时，他就自动拥有该 Schema 下面所有对象的特权。

当用户想添加某一对象特权时，需要先选择对象然后再选择相应的操作(如图 6-3 所示)。常见的操作包括：

- CREATE ANY：是否允许创建对象；
- ALL PRIVILEGES：所有对象特权；
- DROP AND ALTER：废除或修改对象；
- SELECT，INSERT，UPDATE and DELETE：选择，添加，更新与删除；
- INDEX：返回值或对值的引用。
- EXECUTE：执行。

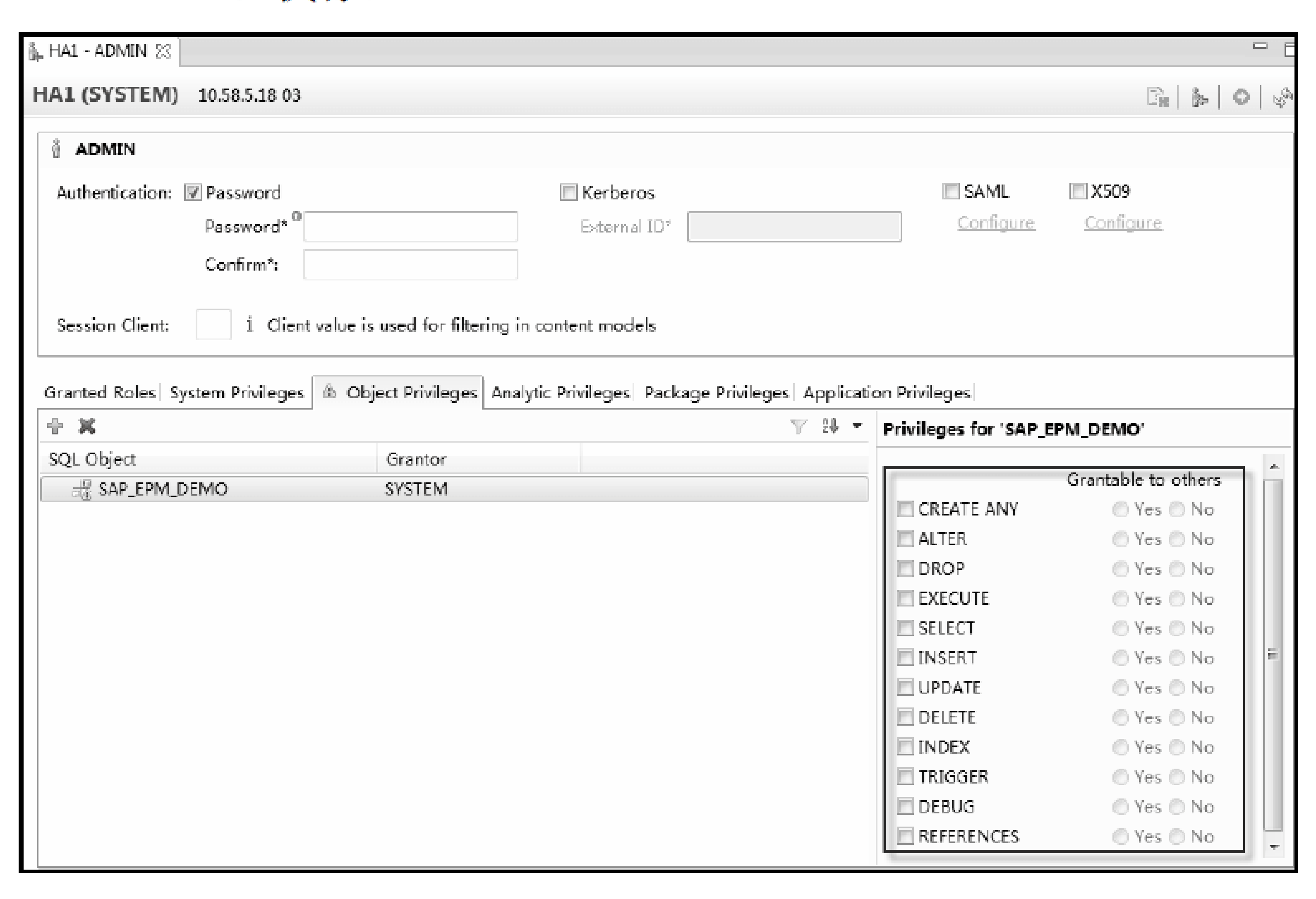


图 6-3

三、分析特权



在 SAP HANA 中，分析特权被用于具体控制哪些用户可以访问哪些分析对象与数据。如果没有相应的分析特权，则以下任何一种分析视图都不能被访问：

- 属性视图
- 分析视图
- 计算视图

SAP HANA 预定义了一个分析特权_SYS_BI_CP_ALL，如果用户被赋予该分析特权，则所有的分析视图与任何数据都能被访问，如图 6-4 所示。

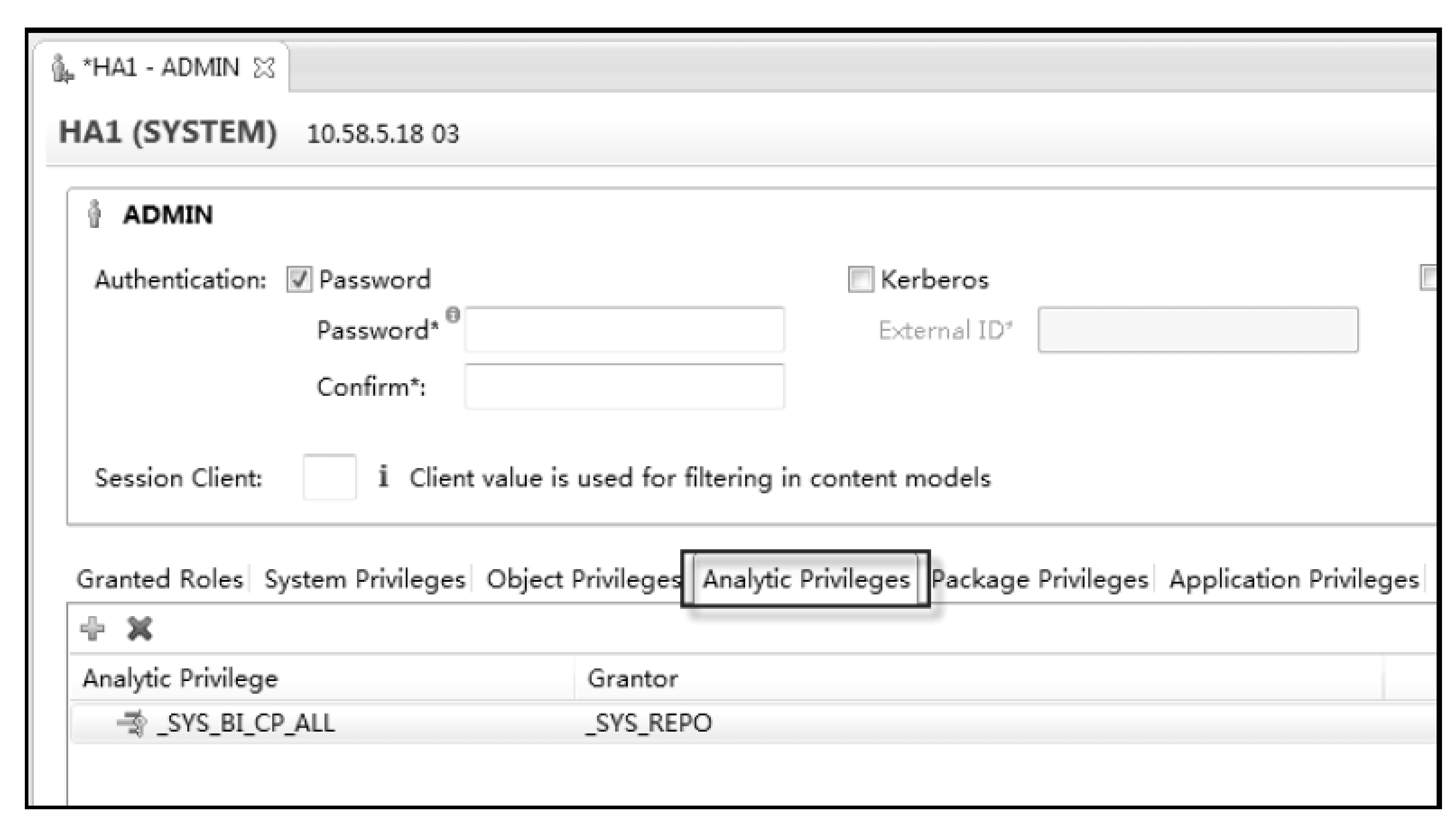


图 6-4

但是，在实际生产系统中，由于分析特权_SYS_BI_CP_ALL 过于强大，我们并不建议使用它。因此，我们往往会需要手动先建立自己的分析特权，然后再赋予不同的用户，在允许访问分析对象的同时，还可以对它设置行级别访问限制，使不同的用户同时访问得到不同的返回数据。在创建自己的分析特权时，我们需要考虑以下两点：

- 对哪些视图允许访问；
- 对哪些视图进行数据限制。

对于如何创建分析特权，我们将在第六节中详细说明。这里我们需要指出的是，对任何一个分析视图，它都是基于某一 Schema 的物理表来实现的。但是，如果一个用户想访问某一分析视图数据(无论数据源涉及的是属性视图、分析视图或者计算视图)，它只需要对该分析视图拥有分析特权即可、并不需要拥有对底层 Schema 的权限。这样做的好处是，用户可以看到被限制的分析数据，而且不能看到源数据表的值，避免了重要信息泄露的漏洞。这个可能现在较难理解，具体原因我们会在



讲解_SYS_REPO 特权时解释。

细心的读者可能会发现，在这里我们所讲述的分析特权数据筛选，都是基于静态的删选条件来实现的。举个例子，如果某公司有 1000 个工厂，我们想要设置限制，使每个工厂的员工只能访问自己工厂的数据来看，这个时候，我们可能需要创立 1000 个不同的分析特权来进行限制。这显然是相当低效且几乎不现实的。其实在 SAP HANA 中，同时还有动态分析特权 Dynamic Analytic Privileges 的概念，它就是为了解决这样的问题而存在的。它能够动态根据不同登录用户来获取不同的权限。不过在本书成稿的时候，它还不支持在 SAP HANA Studio 中创建，而必须通过 SQL 脚本来实现。因此，在本书中，这方面的概念将不做阐述。在图 6-5 中，我们只是给出一个简单例子，请注意在分析删选这一行是通过 Procedure 的方式来实现的。

```
CREATE STRUCTURED PRIVILEGE '<?xml version="1.0" encoding="utf-8"?>
<analyticPrivilegeSchema version="1">
  <analyticPrivilege name="DYN_AP_SALES">
    <cubes>
      <cube name="_SYS_BIC:test.sales/AN_SALES" />
    </cubes>
    <validity> <anyTime/> </validity>
    <activities> <activity activity="read" /> </activities>
    <dimensionAttributes>
      <dimensionAttribute name="test.sales/AT_PRODUCT$PRODUCT_NAME" >
        <restrictions>
           <valueFilter operator="IN">
             <procedureCall schema="PROCEDURE_OWNER"
              procedure="DETERMINE_PRODUCT_FOR_USER" />
           </valueFilter >
           <valueFilter operator="EQ"> <value value="CAR"/>
           </valueFilter>
        </restrictions>
      </dimensionAttribute>
    </dimensionAttributes>
  </analyticPrivilege>
</analyticPrivilegeSchema>';
```

图 6-5



四、包特权

在 SAP HANA 中，所有储存在仓库(Repository)中的对象是通过包的方式来进行分层的(见图 6-6)。对象可以储存在某个包下，而这个包还可以储存在另外一个程序包下面。用户可以被赋予某个包来允许对该包进行相应操作。这里程序包特权可

以向下传递：一旦某一包可以被操作，则所有它的子包都可以被操作。在程序包特权中，可分为两大类:

- 原生包：在本 HANA 系统中创建的程序包；
- 导入包：在其他系统中创建并导入到本系统的程序包。

对于原生包，开发者可以被赋予如下特权：

- REPO.READ：允许读取原生包和导入包中的内容；
- REPO.EDIT_NATIVE_OBJECTS：允许更改原生包对象；
- REPO_ACTIVATE_NATIVE_OBJECTS：允许激活原生包对象；
- REPO.MAINTAIN_NATIVE_PACKAGES：允许更改或删除原生包。

对于导入包，开发者可以被赋予如下权限：

- REPO.EDIT_IMPORTED_OBJECTS：允许编辑导入包对象；
- REPO.ACTIVATE_IMPORTED_OBJECTS：允许激活导入包对象；
- REPO.MAINTAIN_IMPORTED_PACKAGES：允许更改或删除导入包。

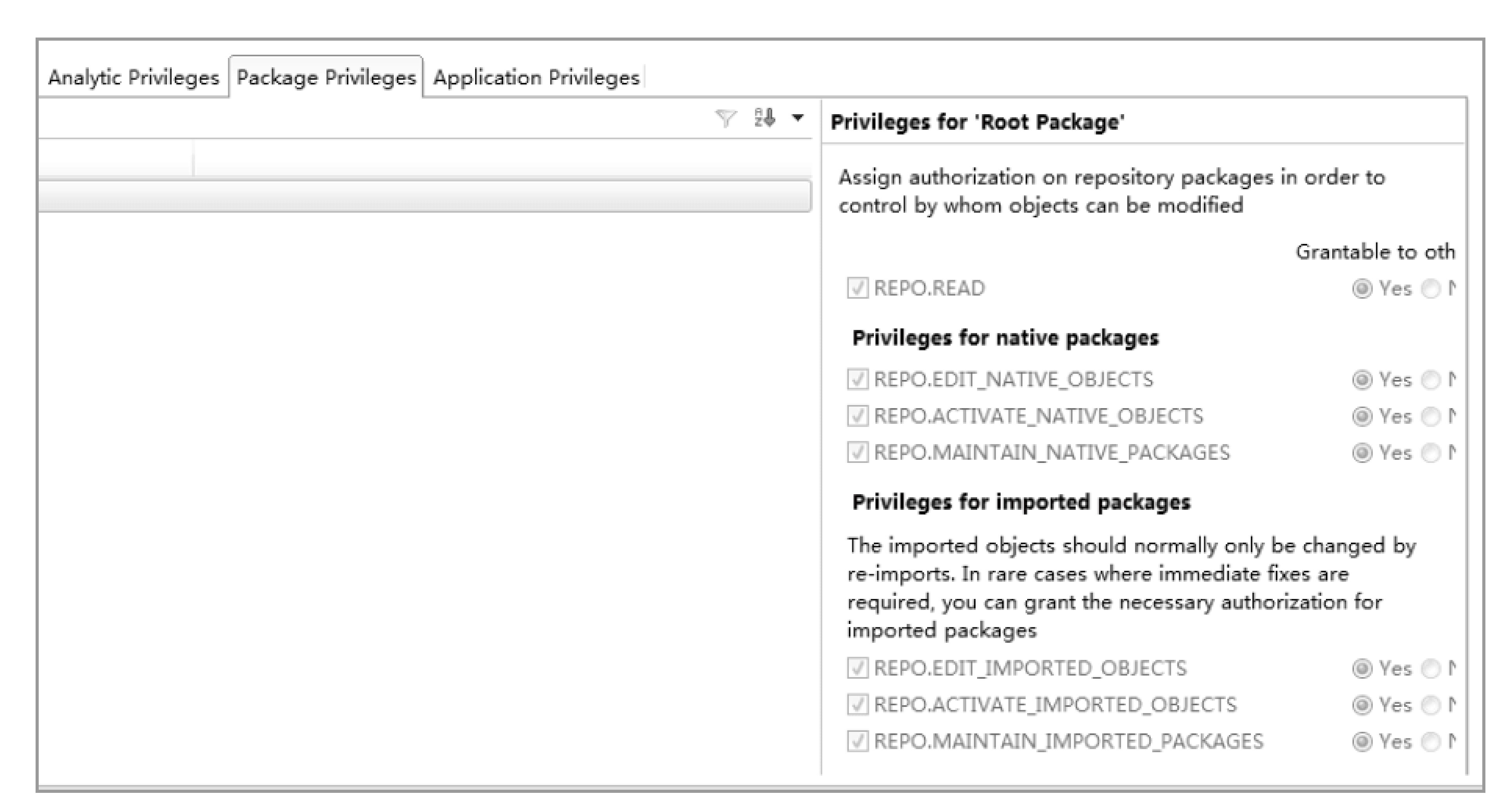


图 6-6



第三节　SAP HANA 角色管理

在 SAP HANA 中，角色是一些特权的集合。它可以被授予用户或者被授予另外的角色。通常，一个角色包含一些特定的特权来实现特定的任务或功能，如：

- 访问 SAP HANA 中分析视图
- 在 SAP HANA 中建模
- SAP HANA 维护人员

在 SAP HANA 中已经有 5 个预定义的角色供开发者快速开发使用。

- MODELING
  - 能够创建分析模型与分析权限，并可以查看所有的视图；
  - 注：不要在生产系统中使用。
- CONTENT_ADMIN
  - 拥有所有 MODELING 拥有的权限，并可以 GRANT 给其他人；
  - 同时可以修改导入的内容；
  - 注：不要在生产系统中使用。
- PUBLIC
  - 每个用户默认拥有的权限，且不能删除；
  - 允许查看系统视图与监控视图。
- MONITORING
  - 允许查看所有元数据(metadata)，当前系统状态与监控视图(monitoring views)。
- SAP_INTERNAL_HANA_SUPPORT
  - 允许查看一些低级别的系统信息，供 SAP HANA 核心开发人员或支持人员使用；
  - 所有权限都是只读的；
  - 不允许访问任何客户的信息。



在本书出版之时，SAP 已经推出了 SAP HANA Cloud。在 SAP HANA Cloud 中除了这几个默认的角色，还有其他针对云的预定义角色。在本文中，只基于传统的 SAP HANA 平台进行阐述。

第四节　SAP HANA 用户管理

在 SAP HANA 内存数据库中，我们可以认为有两种不同的用户类型：

- 数据库用户
  - 这种数据库用户可以和现实中的需要对数据库进行操作的用户相对应。当用户离开组织或是不再需要对数据库进行操作时，所有与该用户相关的对象将被终止。
- 技术用户
  - 他们不与现实中的任何用户对应，这些用户因为某个特定任务而创建。

从技术上讲，这两种用户类型并没有特别大的差异，只是从概念上加以区别并实施不同的权限策略。同角色一样，SAP HANA 已经预定义了一些系统用户用于特殊功能：

- SYSTEM
  - 在 SAP HANA 安装时被创建；
  - 拥有几乎所有系统权限，且不能被取消；
  - 不应在日常应用中使用。
- SYS
  - 是所有内部表、系统视图与监控视图的拥有者；
  - 不能够使用该用户登录系统。
- _SYS_STATISTICS
  - 是在统计服务器中所有对象的拥有者；
  - 统计服务器用于监控 SAP HANA 状态，如性能、资源使用等；
  - 不能够使用该用户登录系统。
- SYS_REPO
  - 是所有设计时对象的拥有者，同时拥有这些对象的激活版本(Activated version)；
  - 拥有 Schema_SYS_BI、_SYS_BIC、_SYS_RT 和_SYS_XS；
  - 不能够使用该用户登录系统。
- _SYS_AFL
  - 是应用功能库(Application Function Libraries)所有对象的拥有者；



  - ➢ 不能够使用该用户登录系统。
- _SYS_DATAPROV
  - ➢ 是所有与数据供应相关的对象拥有者；
  - ➢ 不能够使用该用户登录系统。

第五节 _SYS_REPO 权限管理

一、简介

在这章中，我们会讲述 SAP HANA 所独有的一个授权特点：_SYS_REPO 用户授权。

如前面所述，_SYS_REPO 用户是一个特殊用户，它不能用于登录系统，但它是仓库中所有设计时对象(Design Time Object)的拥有者。让我们通过以下思路来一步步理解这个用户的特性：

- 它是所有设计时对象的拥有者，如属性视图、分析视图和计算视图等。
- 所有的设计时对象必须被激活才能被使用。
- 当我们在前台选择激活对象时，实际是通过_SYS_REPO 来激活对象(而不是我们登录用户)。
- 这些设计时对象也是基于实际 Schema 下面的数据表来建模的。
- _SYS_REPO 默认并没有这些表的权限。
- _SYS_REPO 必须被赋予这些 Schema 或者表的 SELECT 权限(WITH GRANT)才能够成功激活设计时对象。

在激活任何分析视图前，我们需要保证_SYS_REPO 已拥有相应的权限，不然激活这些对象时永远会遇到权限缺失错误(连 SYSTEM 用户激活都不能成功)。



二、对激活对象授予或回收权限

只有_SYS_REPO 拥有所有设计时对象相关的权限，因此，只有这个用户才能赋予它所拥有的权限给其他人。但是，这个用户并不能登录系统，因此我们不得不使用其他方法来完成该操作。在 SAP HANA 中，提供了一些预定义的存储过程来实

现上述的功能，如表 6-1 所示。

表 6-1

对象	存储过程
建模对象，如计算视图等	GRANT_PRIVILEGE_ON_ACTIVATED_CONTENT REVOKE_PRIVILEGE_ON_ACTIVATED_CONTENT
Schema	GRANT_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT REVOKE_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT
分析特权	GRANT_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT REVOKE_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT
应用特权	GRANT_APPLICATION_PRIVILEGE REVOKE_APPLICATION_PRIVILEGE
角色	GRANT_ACTIVATED_ ROLE REVOKE_ACTIVATED_ ROLE

我们可以使用“CALL”语句调用存储过程来为用户赋予权限，例如：

- CALL "_SYS_REPO"."GRANT_ACTIVATED_ROLE"('sap.hana.democontent.epm.data::model_access','HANA_USER');
- CALL "_SYS_REPO"."REVOKE_ACTIVATED_ROLE"('sap.hana.democontent.epm.data::model_access','HANA_USER');
- CALL "_SYS_REPO"."GRANT_ACTIVATED_ANALYTICAL_PRIVILEGE"('"sap.ecc.fica/AN_TEST"','HANA_USER');

第六节　常见任务

在介绍完 SAP HANA 用户管理的基本概念之后，在这里我们通过几个现实业务中的常见任务来进一步加深我们的理解。图 6-7 中我们列出了一些在企业中常见的用户与权限管理流程。



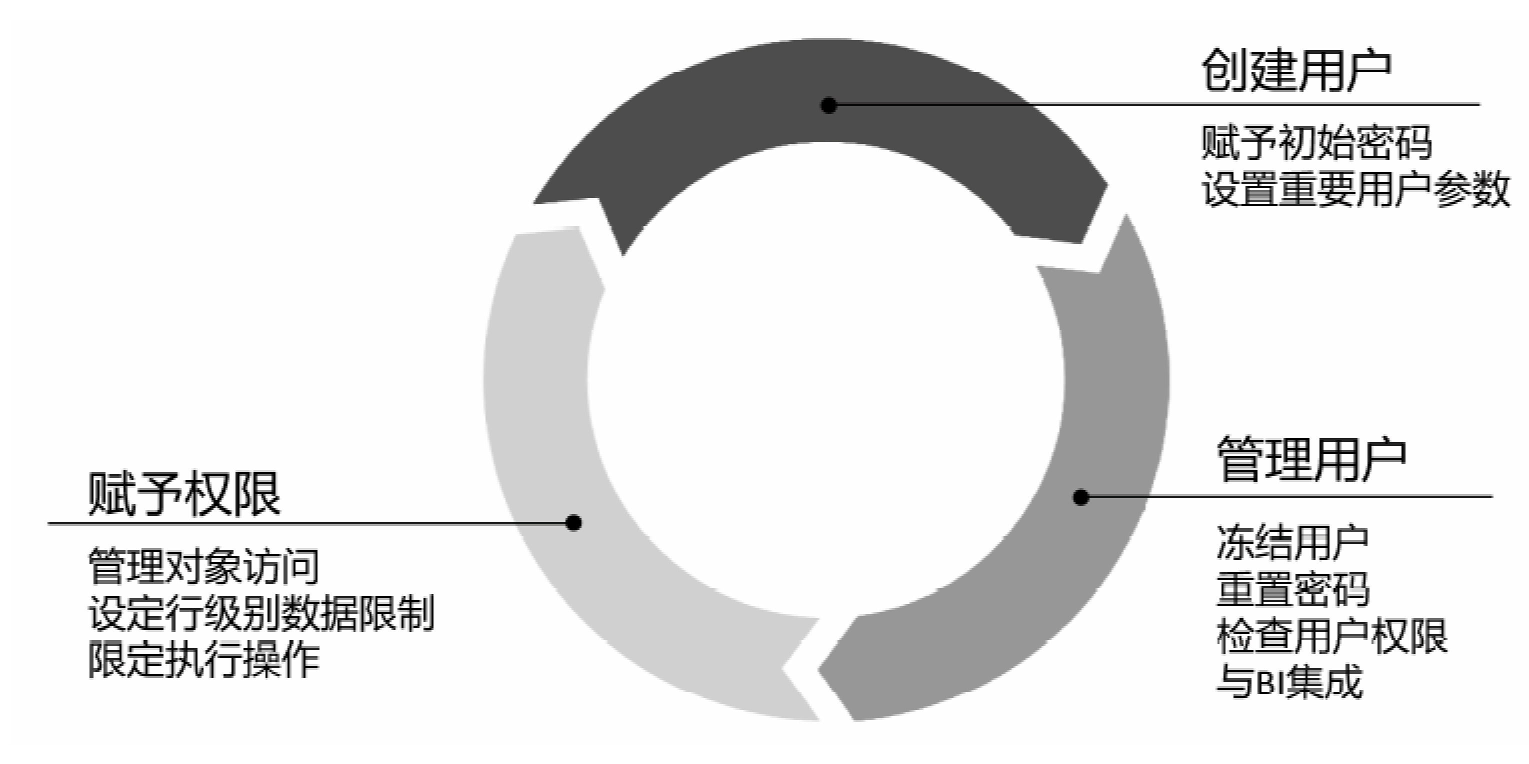


图 6-7

一、创建建模用户

本例将介绍如何创建一个开发用户用于 SAP HANA 建模。

(1) 打开 SAP HANA Studio，在“SAP HANA Systems”视图中定位到 <(SYSTEM)>→“Security”→“Users”(见图 6-8)。

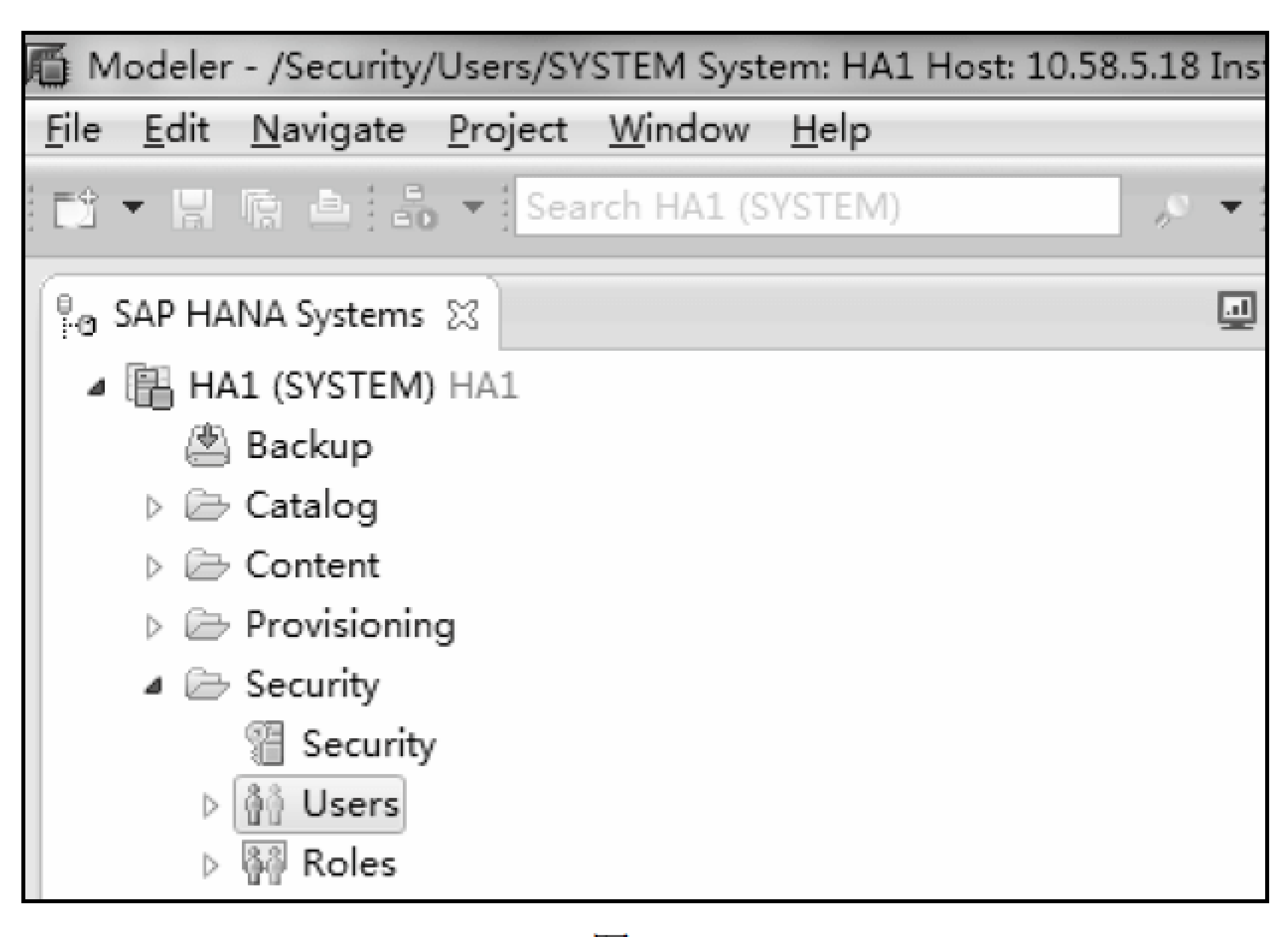
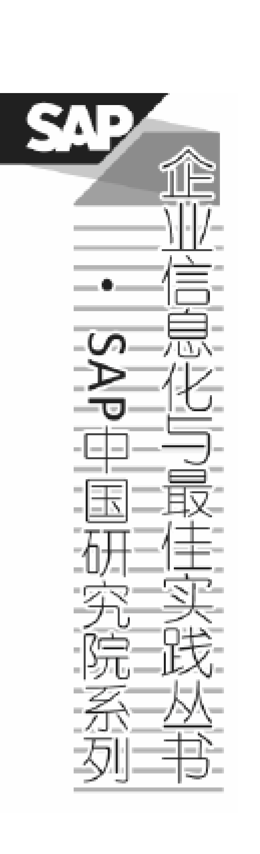


图 6-8



(2) 右击“Users”节点，选择“New Users”选项(见图 6-9)。

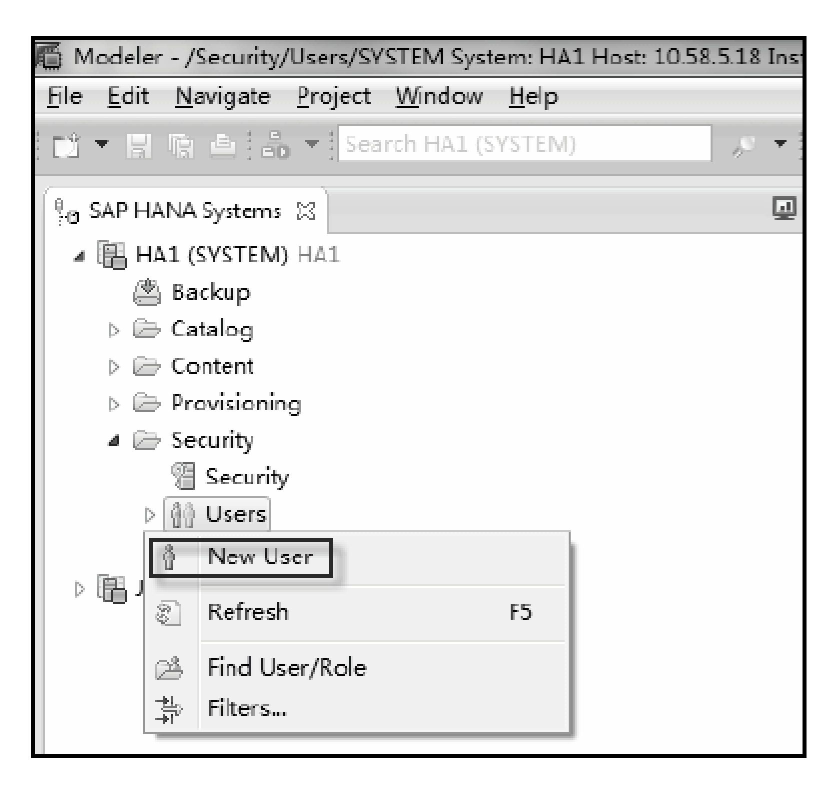


图 6-9

(3) 输入用户名与密码。请注意密码需包含大小写字母和数字(见图 6-10)。

*HA1 - New User
HA1 (SYSTEM) 10.58.5.18 03
New User
User Name*: XXXXXXX
Authentication: Password (Password must have at least eight characters and meet the rule: "upper-case", "lower-case", "digits")
Password*: •••••••• External ID*
Confirm*: ••••••••
Session Client: Client value is used for filtering in content models

图 6-10

(4) 在“Granted Roles”标签，选择添加(+)，如图 6-11 所示。

*HA1 - New User
HA1 (SYSTEM) 10.58.5.18 03
New User
User Name*: XXXXXXX
Authentication: Password Kerberos
Password*: •••••••• External ID*
Confirm*: ••••••••
Session Client: Client value is used for filtering in content models
Granted Roles | System Privileges | Object Privileges | Analytic Privileges | Package Privileges | Application P
Role Grantor Details

图 6-11

(5) 在弹出窗口中的查找字段输入“MODELING”并单击“OK”按钮(见图 6-12)。

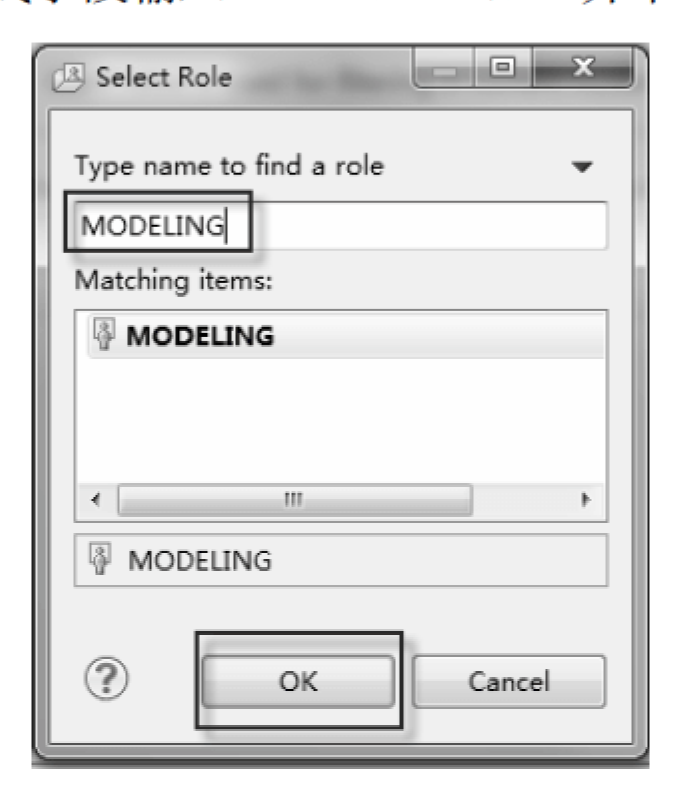


图 6-12

(6) 切换到“Object Privileges”标签，为此用户添加所需要的 Schema 相关的权限。例如我们要为此用户添加本书中需要用的“SAP_HANA_EPM_DEMO”Schema，则可以选择添加(+)，在弹出的窗口中输入“SAP_HANA_EPM_DEMO”，单击“OK”按钮(见图 6-13)。

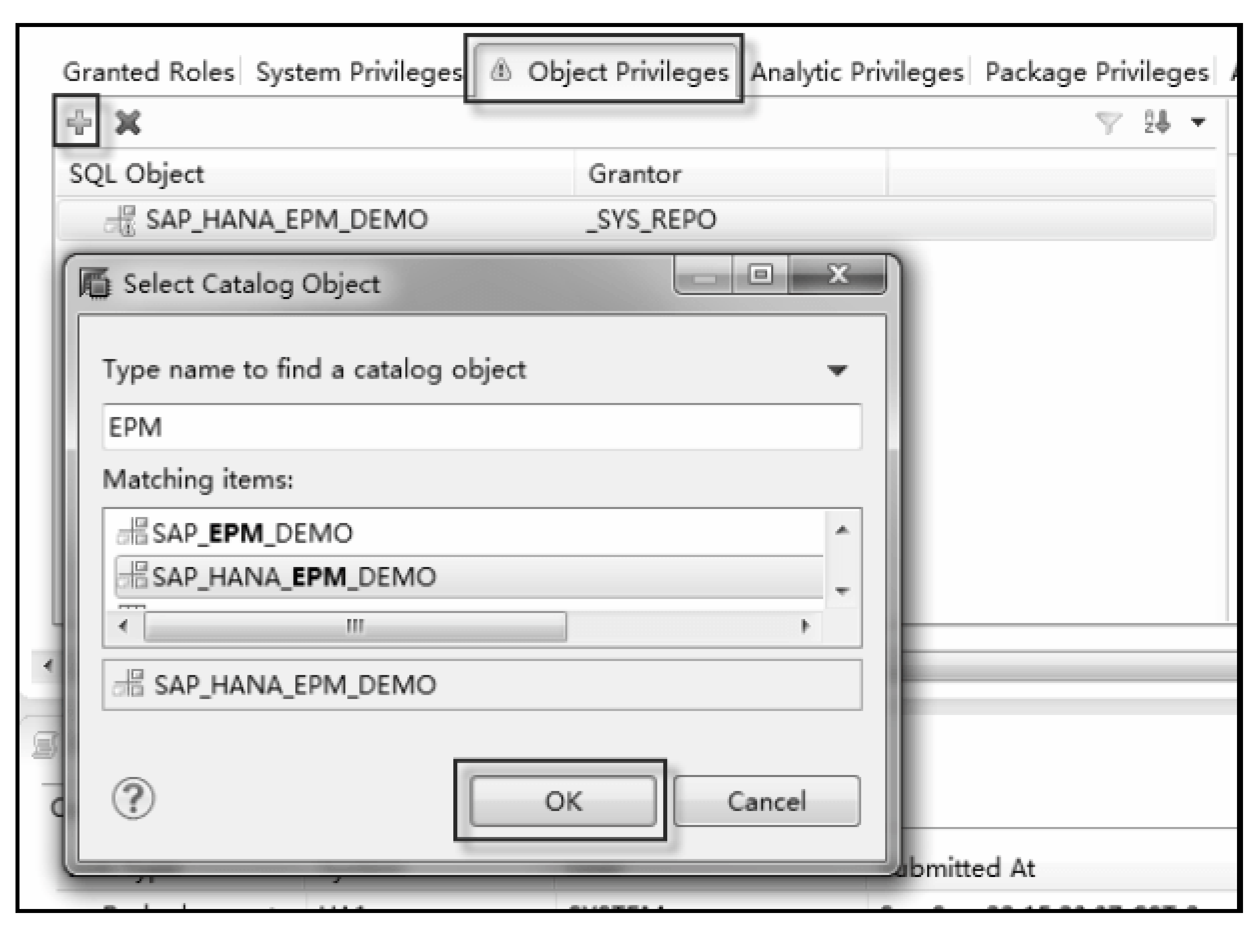


图 6-13

(7) 添加完新的 Schema 后，我们需要在右侧的区域中为该用户赋予详细的权限，请根据实际情况来赋予(见图 6-14)。

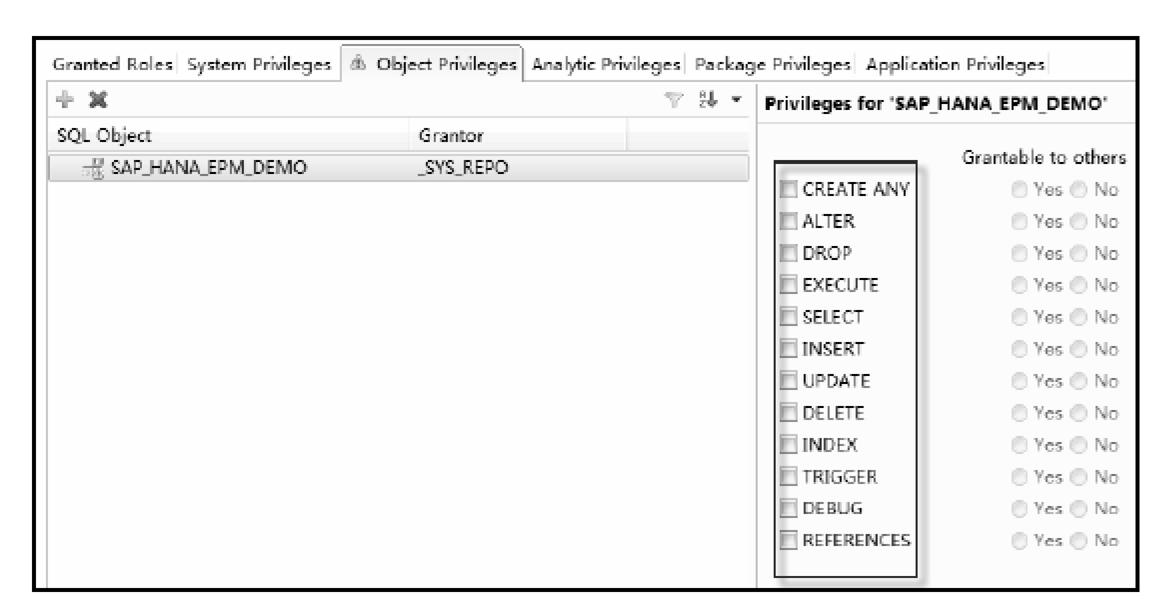


图 6-14

(8) 选中“System Privileges”标签，选择你需要的系统权限，如 USER ADMIN 等。若此用户的用途只用于 SAP HANA 建模，则可以不用在此处做任何配置，留空即可(见图 6-15)。

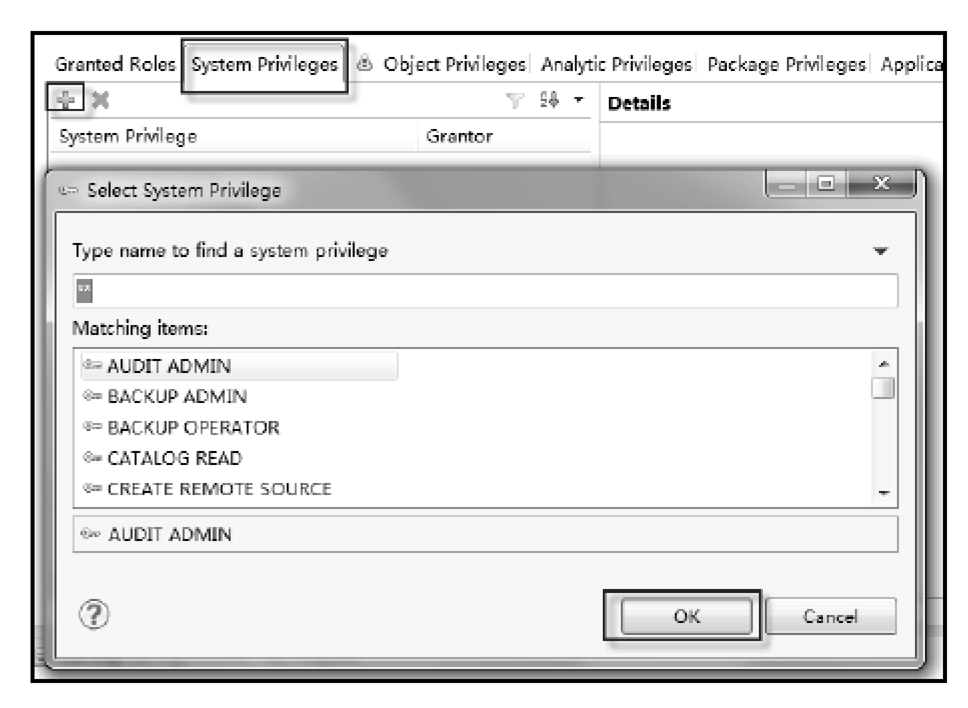


图 6-15

(9) 单击“Deploy”按钮或直接按 F8 键来部署和激活用户(见图 6-16)。

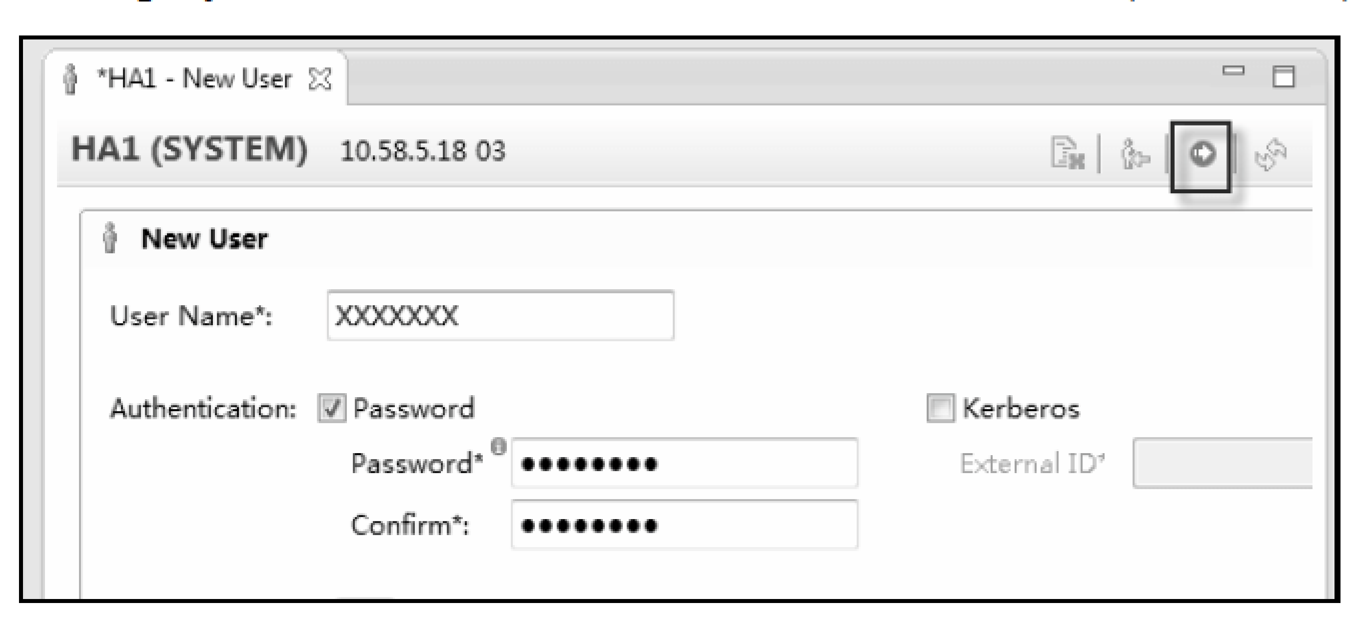


图 6-16

此时我们已经创立了一个拥有开发权限的用户。但在系统中_SYS_REPO 是所有可激活对象的拥有者，所以为了开发需要，我们还需要对用户_SYS_REPO 赋予相应权限。

(10) 在左边系统列表中，右键选择“Add Additional User...”(见图 6-17)。

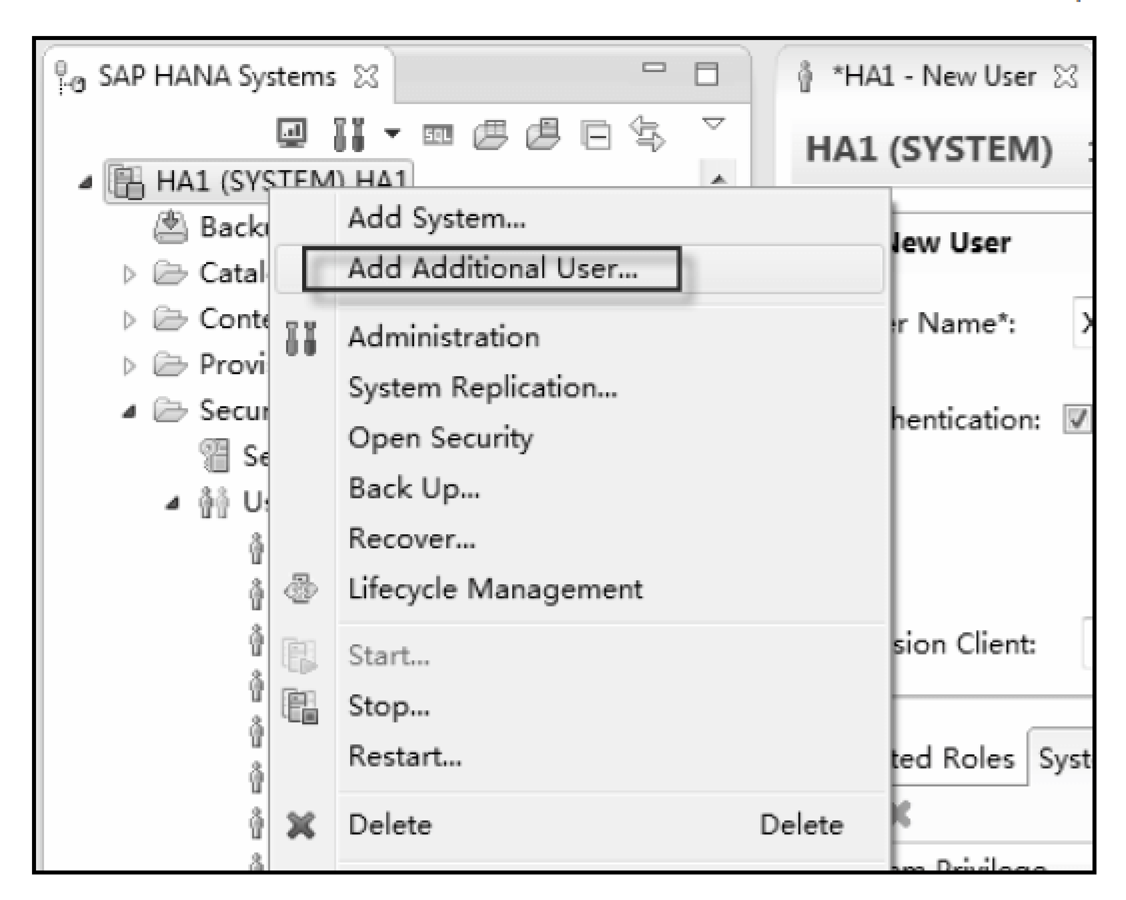


图 6-17

(11) 使用刚才创建并激活的用户名及密码登录(见图 6-18)。



图 6-18

(12) 在“SAP HANA Systems”视图中定位到<SYSTEM>→“Security”→“Users”，展开“Users”并找到“_SYS_REPO”用户，双击(见图 6-19)。

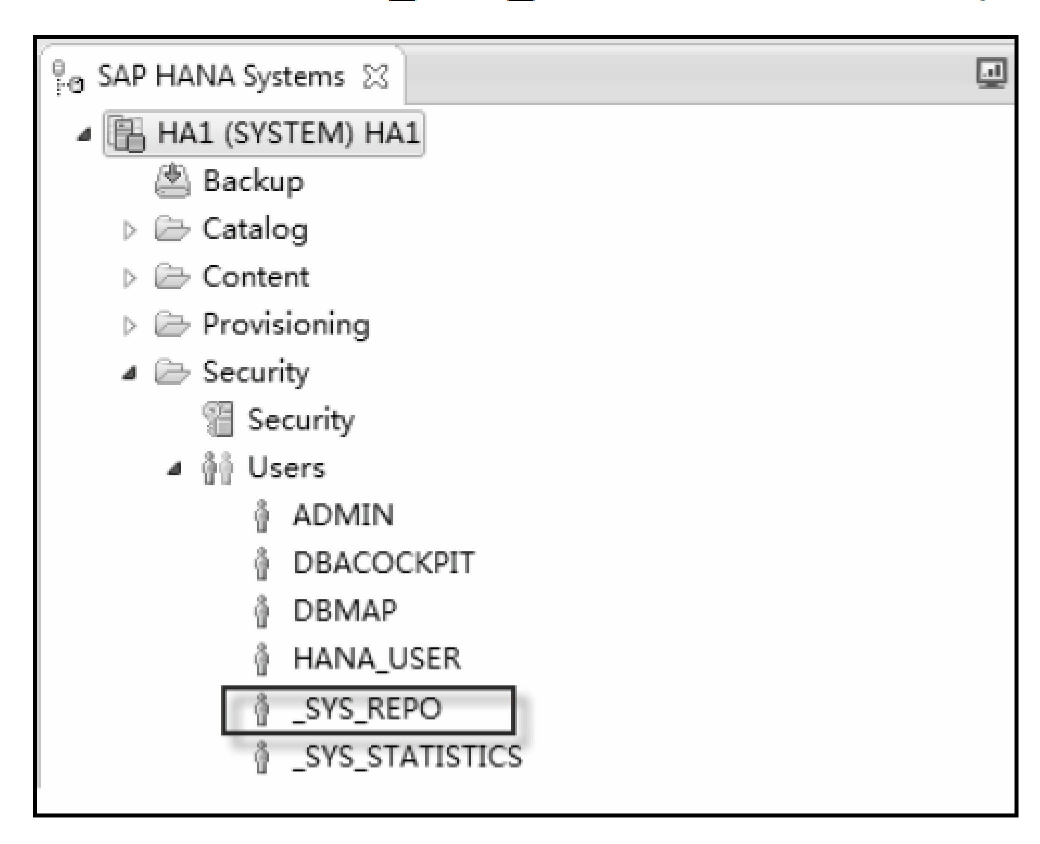


图 6-19

(13) 选中“Object Privileges”标签，对想做分析的 Schema 添加 SELECT 权限，并选中“Grantable to others”(见图 6-20)。

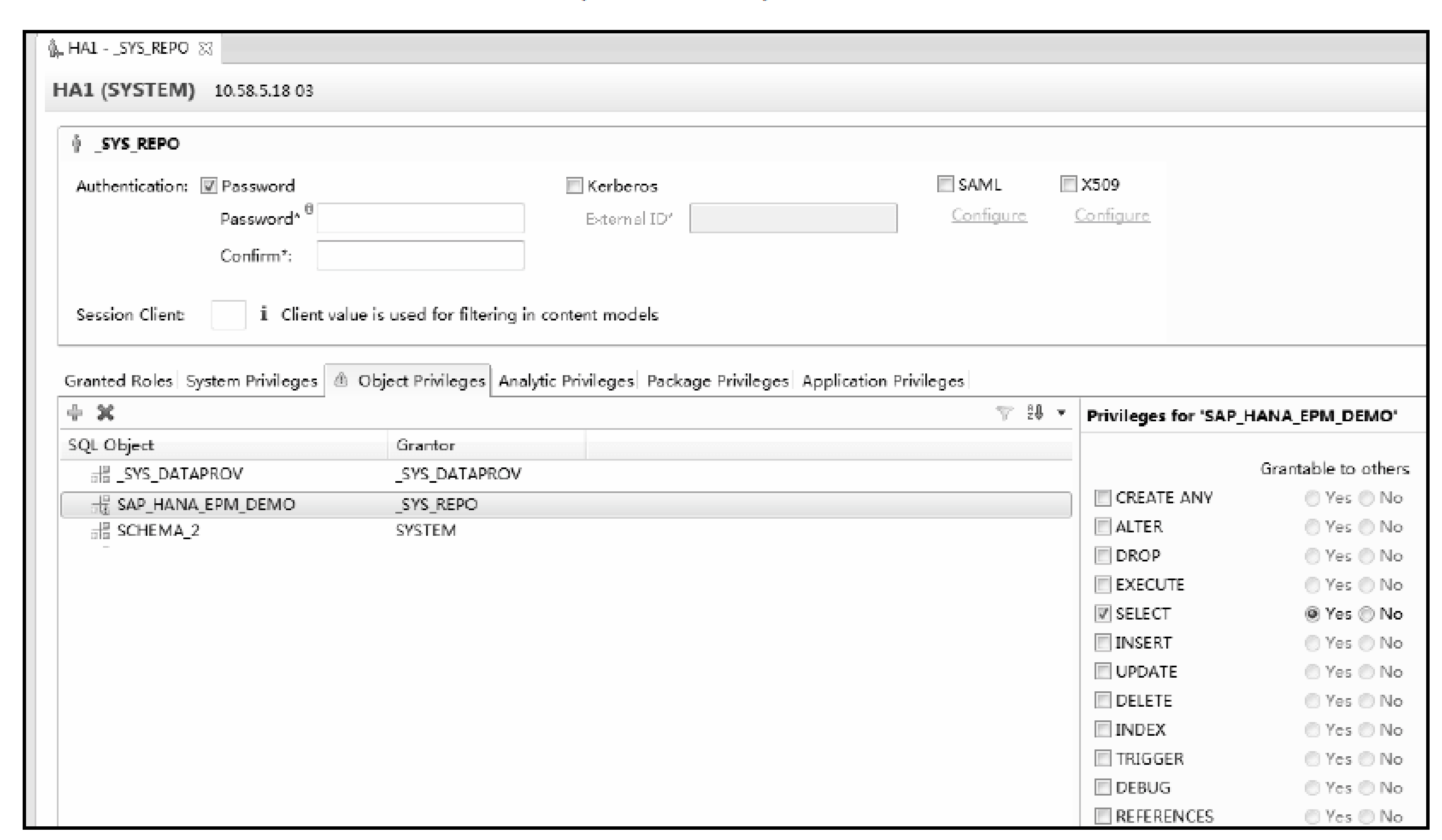


图 6-20

(14) 单击“Deploy”按钮或直接按 F8 键来部署和激活用户(见图 6-21)。



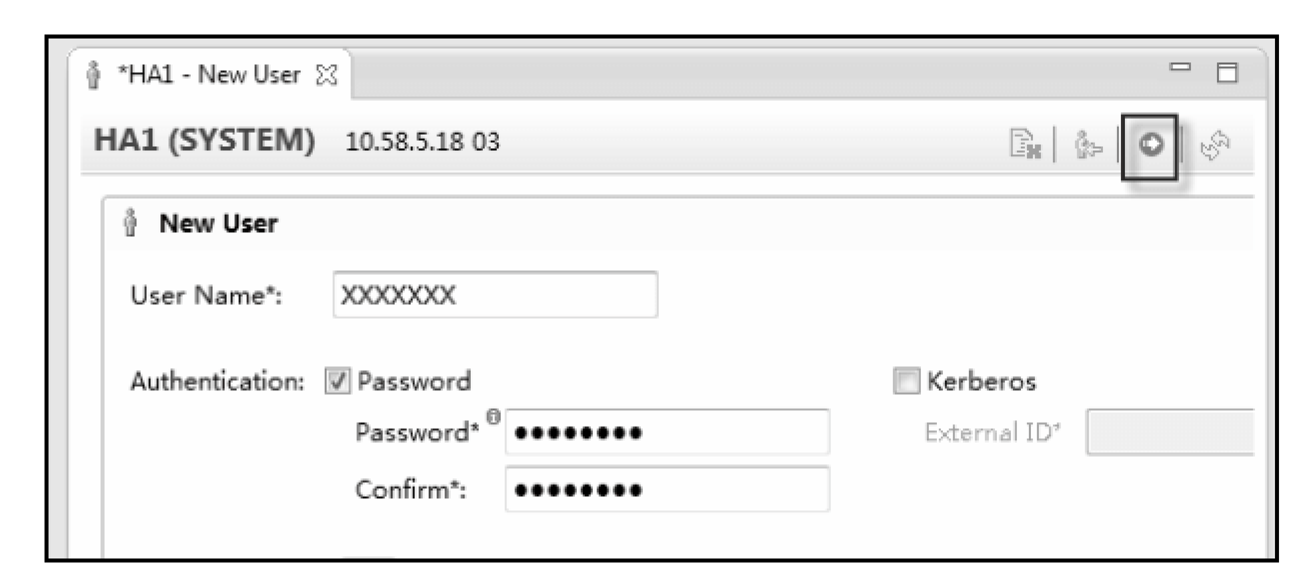


图 6-21

(15) 在创建完用户之后，如果想检查用户所拥有的特权列表，可以通过 SQL 查询来实现：

```
SELECT * FROM EFFECTIVE_PRIVILEGES WHERE USER_NAME = <USER_NAME>;
```

二、用户密码设定与重置

在创建完用户之后，有时候由于特定的业务需要，我们会更改一些用户的安全策略以确保业务不会因意外而中断，如用户被锁、密码被重置等情况。在 SAP HANA Studio 中，我们可以通过配置用户密码策略来完成这一需求，具体操作如下。

(1) 打开 SAP HANA Studio，在“SAP HANA Systems”视图中双击系统图标(见图 6-22)。



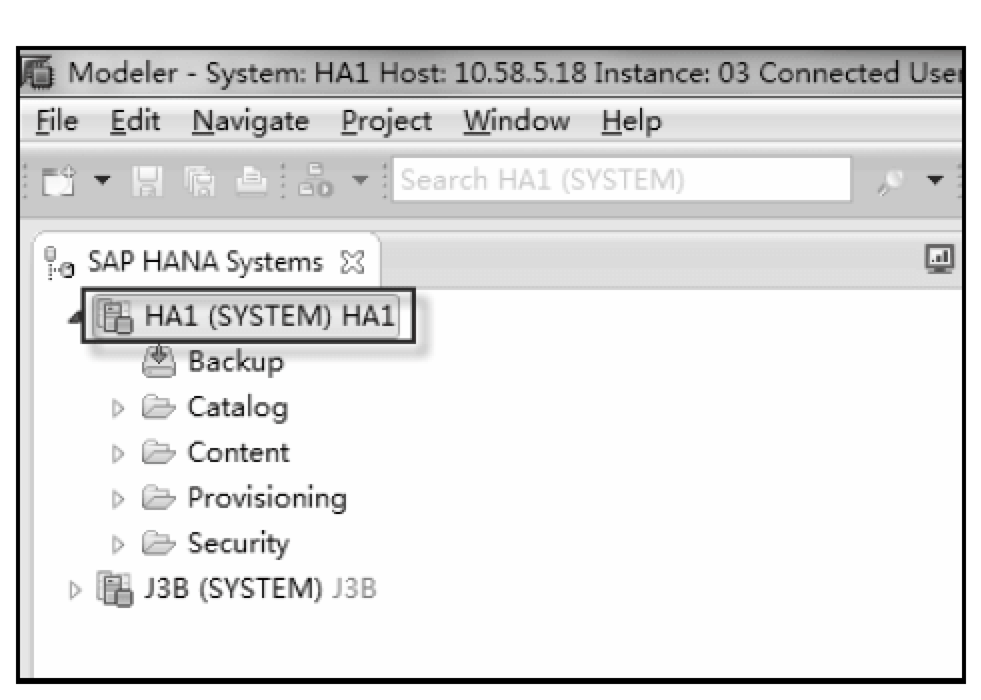


图 6-22

(2) 在右侧的系统配置视图中，选中“Configuration”标签(见图 6-23)。

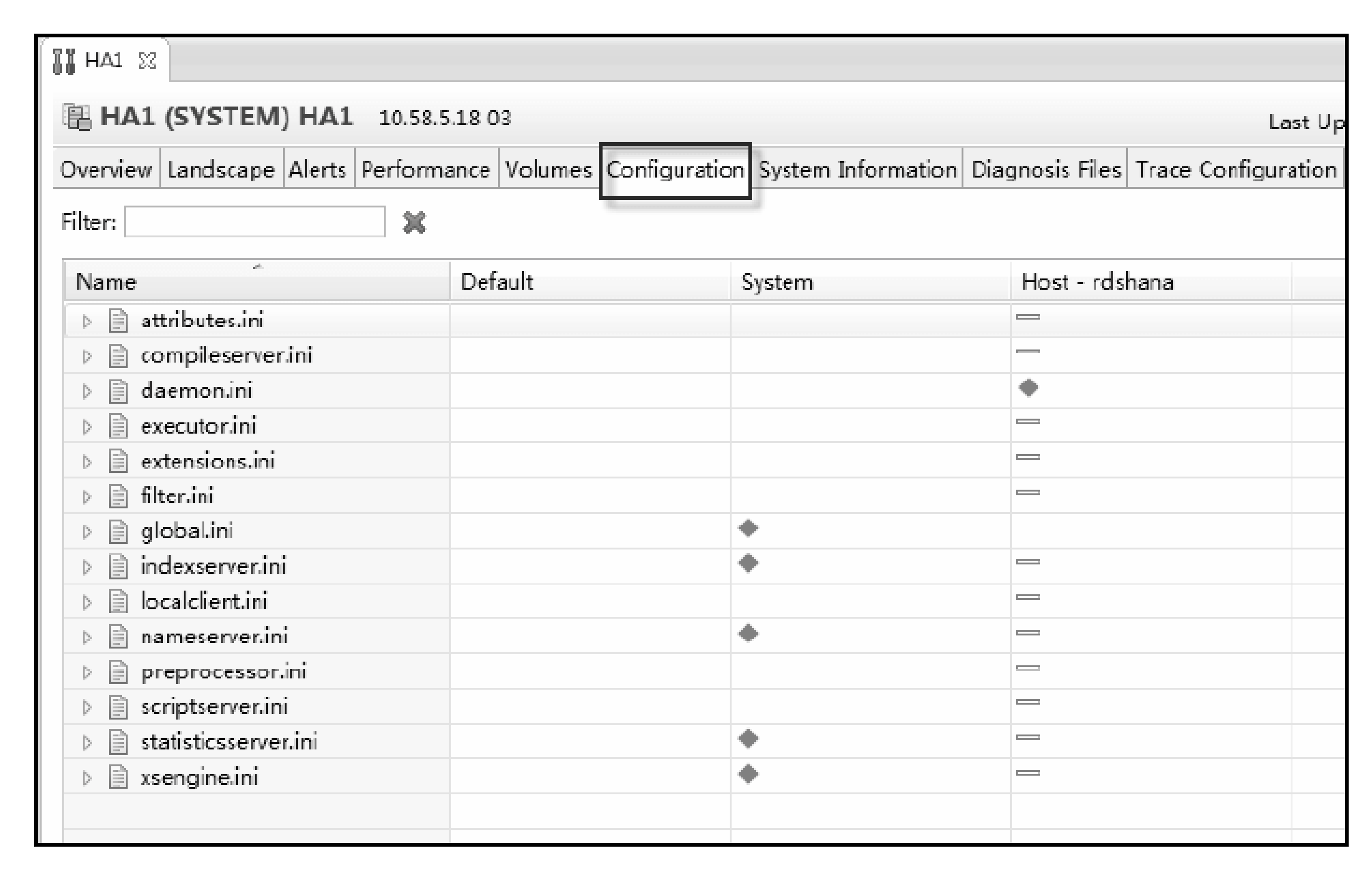


图 6-23

(3) 依次展开“indexserver.ini”→“password policy”后即可配置符合需求的密码策略(见图 6-24)。

[ ] password policy	true
force_first_password_change	true
last_used_passwords	5
maximum_invalid_connect_attempts	6
maximum_password_lifetime	182
maximum_unused_inital_password_lifetime	28
maximum_unused_productive_password_lifetime	365
minimal_password_length	8
minimum_password_lifetime	1
password_expire_warning_time	14
password_layout	A1a
password_lock_time	1440

图 6-24

(4) 在上图中我们看到，对于普通用户，密码都是有生命周期的。如果我们希望对一些特殊的技术用户永远不必更改密码，可以执行 SQL 语句：

```
ALTER USER <USER_NAME> DISABLE PASSWORD LIFETIME;
```

三、分析特权：权限控制

在很多时候，我们需要对一个用户进行限制，使其不能够读取所有的表数据。如一个区域销售经理只能看到属于自己区域的销售数据。在 SAP HANA 中，我们可



以通过分析特权(Analytic Privilege)来实现。请注意，在这里分析特权只能对分析视图进行限制，也就是说无法对原始表进行限制。

接下来我们介绍一下如何在 SAP HANA Studio 里面创建分析特权。

(1) 打开 SAP HANA Studio，在“SAP HANA Systems”视图中定位到<SAP HANA HOST>→“Content”→<需创建分析特权的包>。你可以按照图 6-25 所示的路径打开 EPM 实例中的包来进行练习。

在需要创建分析特权的包上面右击鼠标，选择“New”→“Analytic Privilege”。

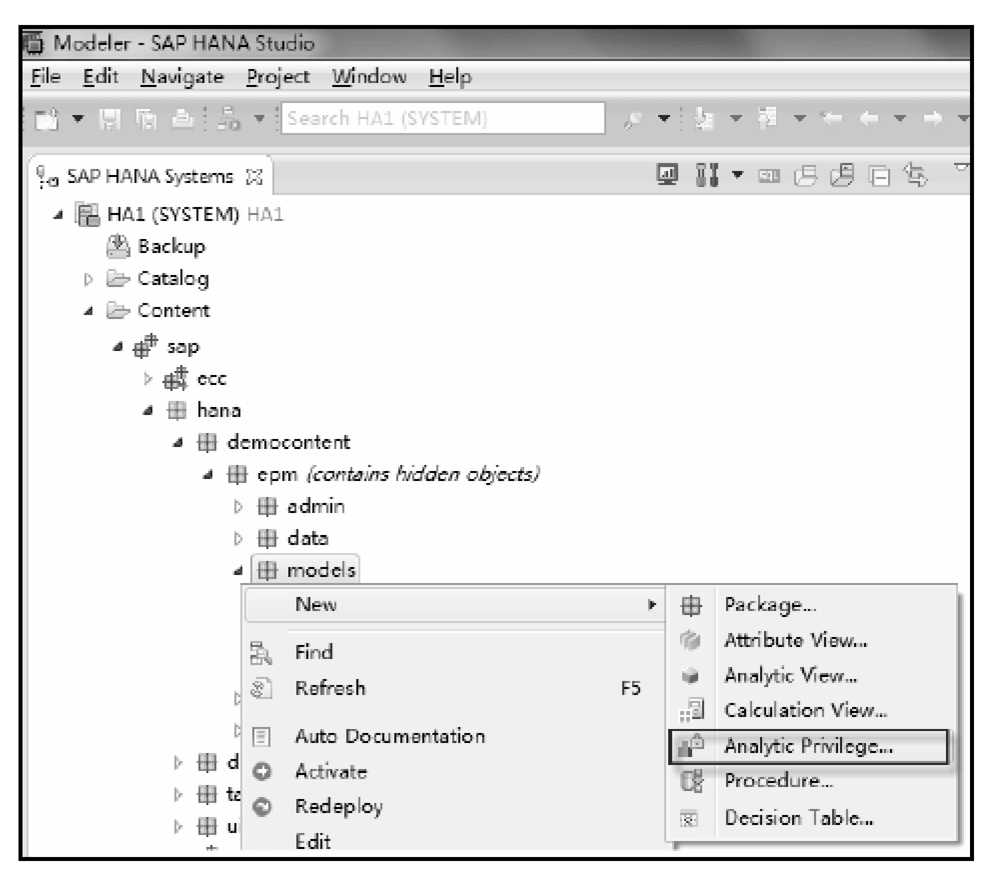


图 6-25

(2) 输入分析特权名称及描述，选中“Create New”并单击“Finish”按钮(见图 6-26)。

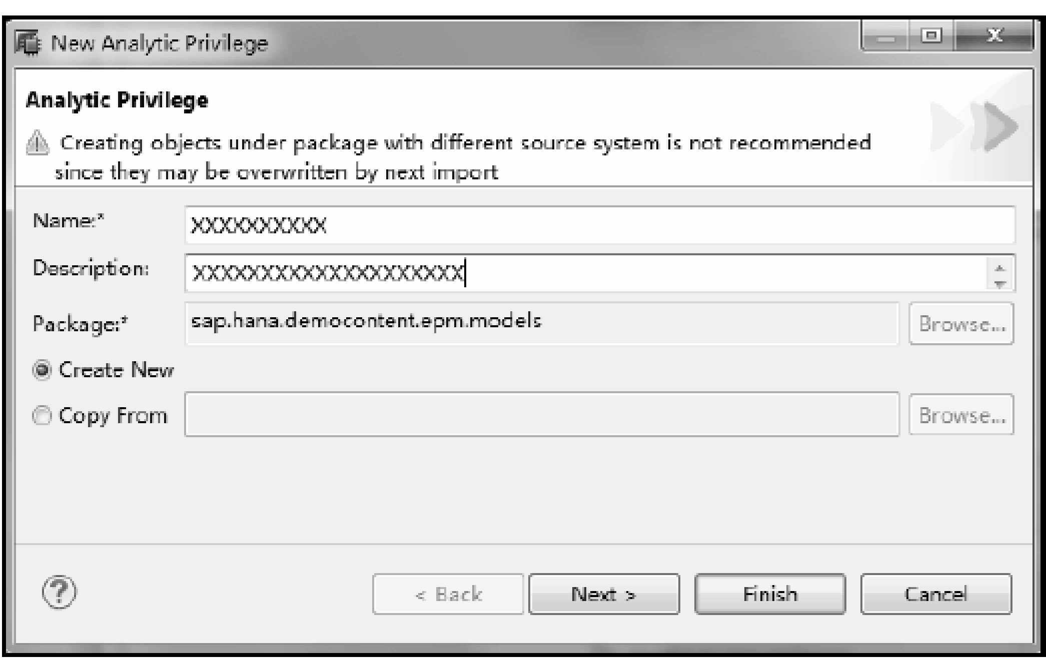


图 6-26



(3) 你将会看到如图 6-27 所示的分析特权配置画面。在此画面中“Reference Models”区域点击“Add…”按钮。

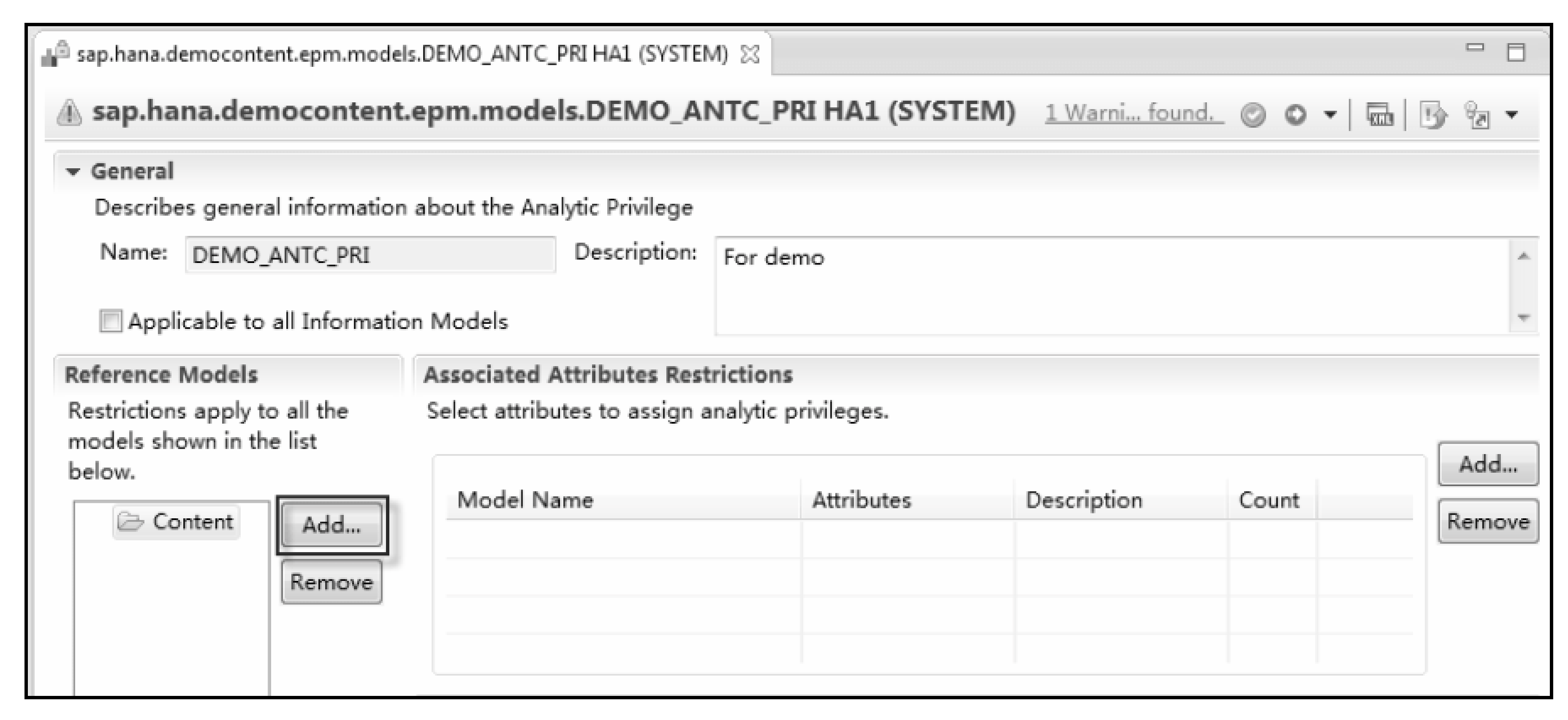


图 6-27

(4) 在随后弹出的“Information Model”屏幕列表中选中你要添加特权的视图(见图 6-28)，本例中我们依旧使用 EPM 实例中的分析视图，如“AN_SALES_OVERVIEW”。

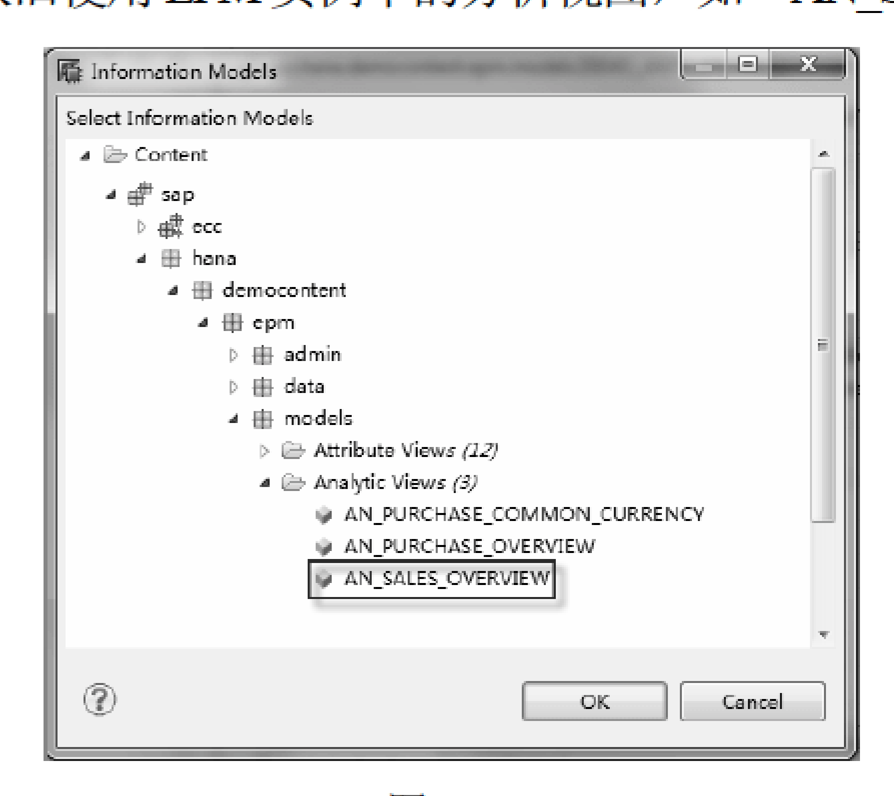


图 6-28

(5) 添加完成后，我们在右边的“Associated Attributes Restrictions”区域通过点击“Add…”按钮来选择需要限制的属性，例如我们想对“Posting_date”做限制(图 6-29)。

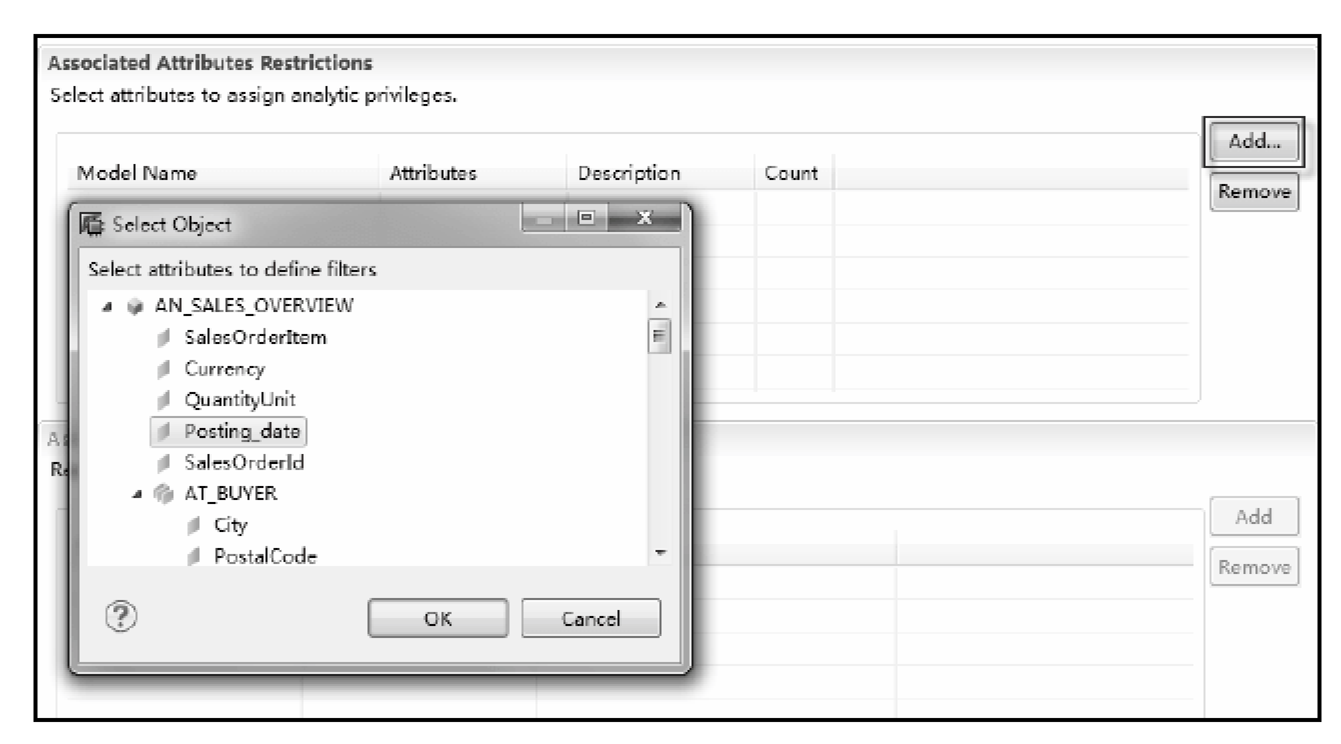


图 6-29

(6) 选择完需要限制的属性后，我们需要在“Assign Restrictions”区域为选定的属性设置公式。例如我们需要限制“Posting_date”在某一特定的时间段，则可以输入如图 6-30 所示的公式。



图 6-30



至此，我们能很清楚地看出这个分析特权定义的是在“AN_SALES_OVERVIEW”这个分析视图中，对“Posting_date”这个属性定义了只在 2012-01-01 到 2012-12-31 时间范围内取值。

(7) 最后单击保存并激活按钮来激活分析特权(见图 6-31)。

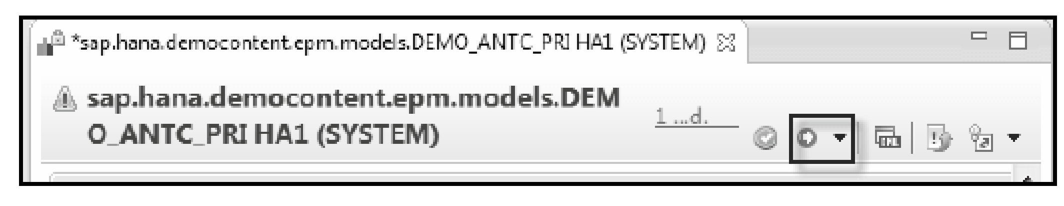


图 6-31

四、创建只读用户

很多情况下，我们可能只需要用户对系统的操作拥有读取的权限即可，这种情况下我们就要创建拥有只读权限的用户来实现这一功能。考虑到可重复利用性，我

们通常先创建一个拥有只读权限的角色，然后就可以把这个角色赋予给多个用户，即可非常方便地创建只读用户。

(1) 在我们创建新用户前，我们先来创建一个角色。打开 SAP HANA Studio，在“SAP HANA Systems”视图中定位到<SAP HANA HOST>→“Security”→“Roles”(见图 6-32)。

(2) 右击“Roles”节点，选择“New Role”选项(见图 6-33)。

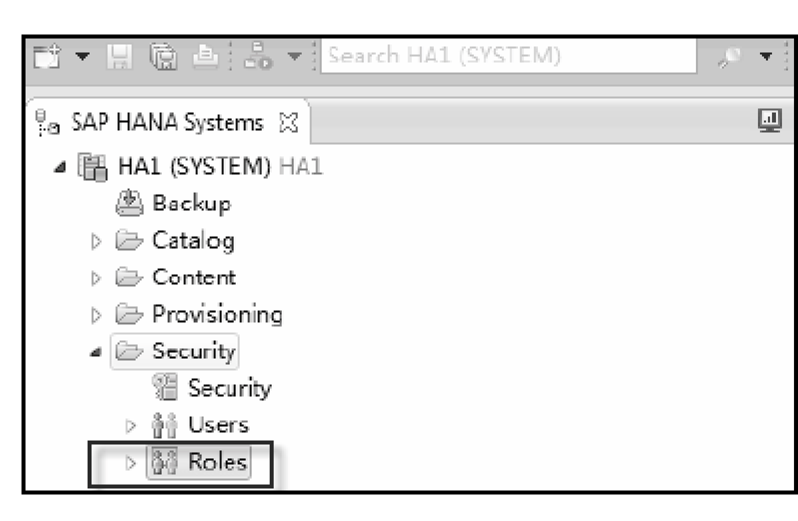


图 6-32

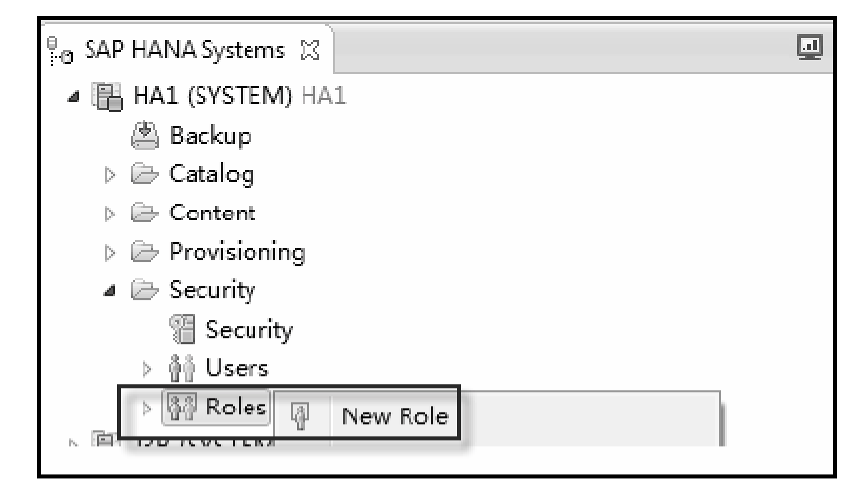


图 6-33

(3) 为新的角色取名(见图 6-34)。

(4) 在“Object Privileges”标签下添加如图 6-35 所示的对象。

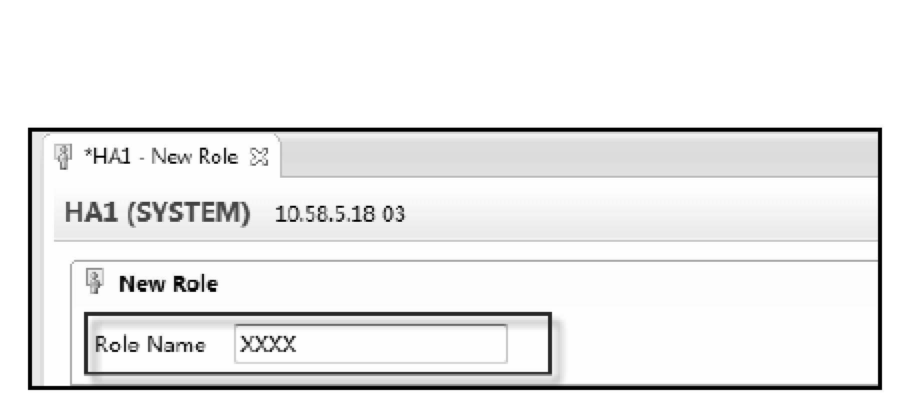


图 6-34

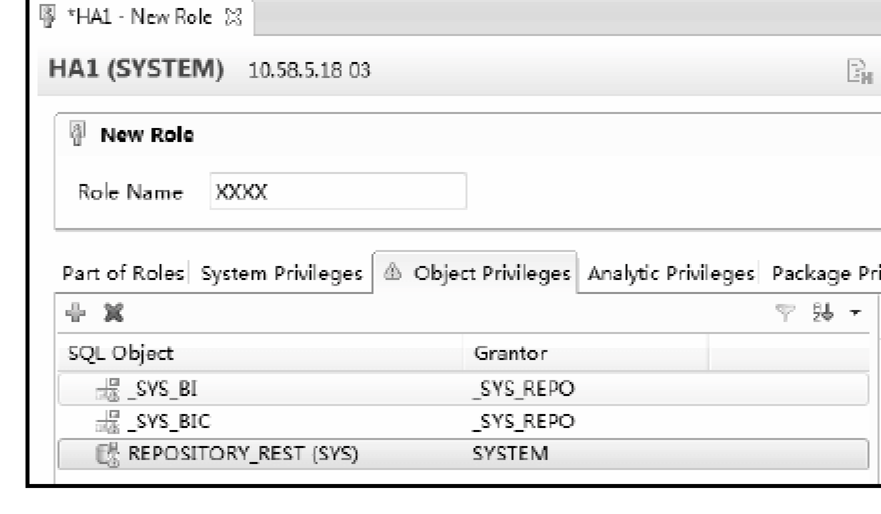


图 6-35

(5) 选中“_SYS_BI”，在右侧特权中选中“SELECT”(见图 6-36)。

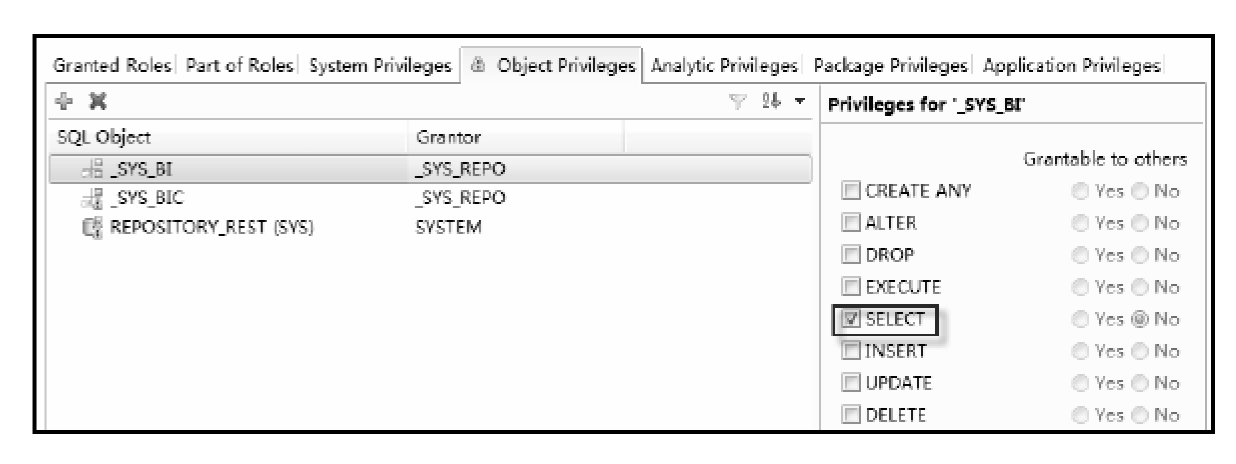


图 6-36

(6) 选中“_SYS_BIC”，在右侧特权中选中“SELECT”(见图 6-37)。

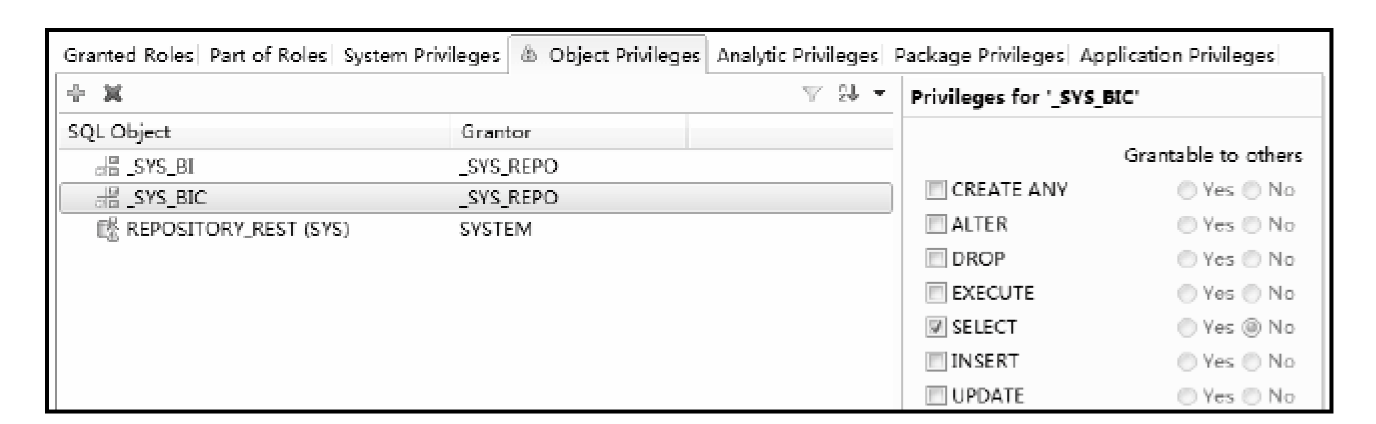


图 6-37

(7) 选中“REPOSITORY_REST(SYS)”，在右侧特权中选中“EXECUTE”(见图 6-38)。

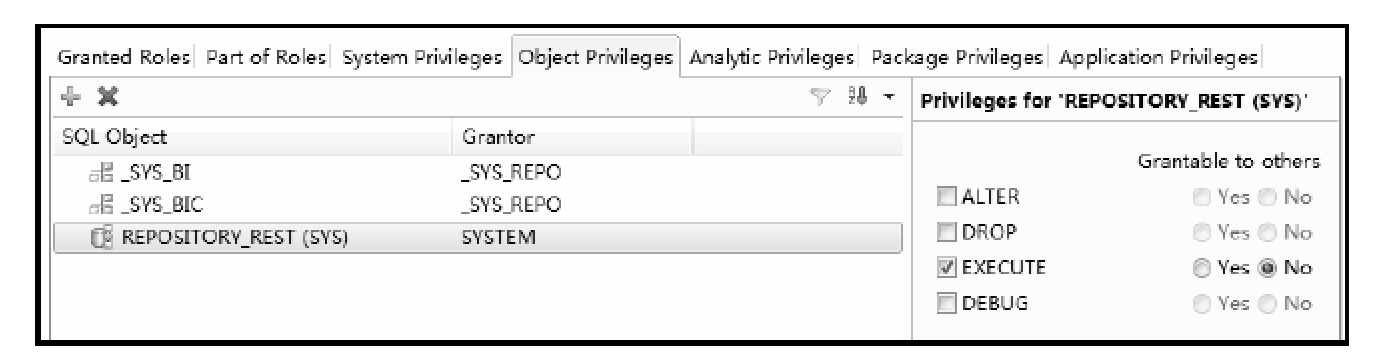


图 6-38

(8) 选中“Deploy”或者直接按 F8 键激活角色(见图 6-39)。

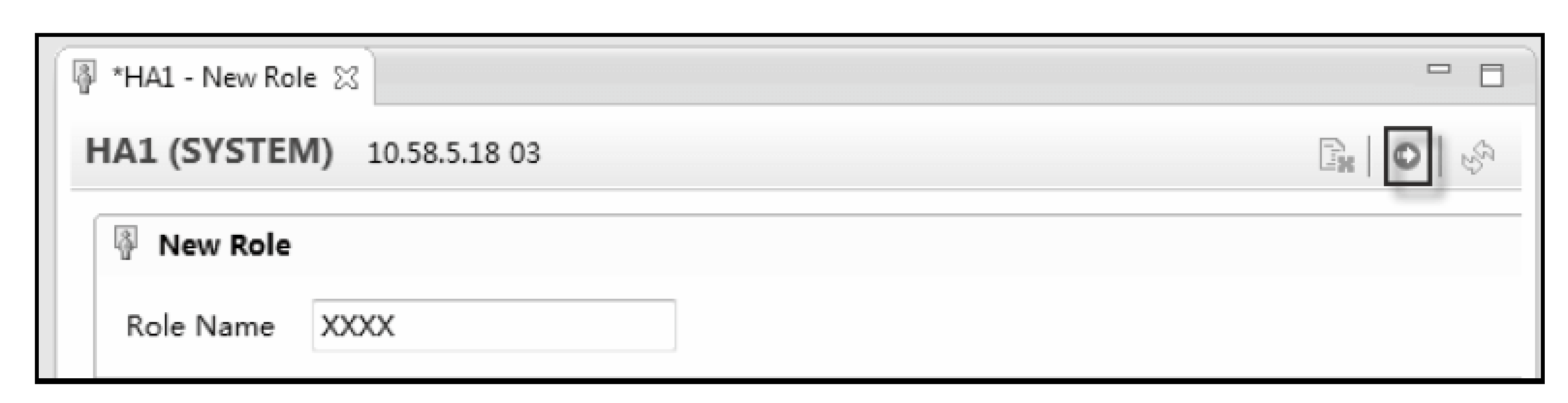


图 6-39

(9) 将此角色赋予需要只读权限的用户。



第七节　常 见 问 题

最后，让我们看一下在用户管理中常见的一些问题。

- 为什么我使用 SYSTEM 用户却没有办法对我的 Schema 授予 SELECT 权限？

回答：

- ➢ 虽然 SYSTEM 已经是超级用户，但是也不能对所有 Schema 进行操作。
- ➢ 请使用 Schema 所有者登录对 SYSTEM 用户授予相应 GRANT 权限，这样 SYSTEM 才能对你的 Schema 进行相应操作。

- 为什么我每次我想激活对象，系统总是告诉我没有权限？

回答：

- ➢ _SYS_REPO 是所有的设计时对象的拥有者，在激活过程中，会由它来激活对象，因此需要授予_SYS_REPO 相应的 Schema 权限才能激活。而且，并不是仅仅对 Schema 的 SELECT 权限就足够，还需要对 Schema 的 GRANT 权限。
- ➢ 激活对象后，对象会存在于_SYS_BIC 的 Schema 下面，我们还需要访问_SYS_BIC 的权限。

- 当我选择视图时，为什么所得到的结果并不是我想要的？

回答：

- ➢ 检查是否有分析权限授予了这个用户。
- ➢ 作为测试用，也可以先授予 CONTENT_ADMIN 或者 MODELER 的角色给用户，这些角色已经默认拥有了“看见所有”的权限。

- 当我想激活一个存储过程时，系统告诉我没有相应权限。但是存储过程中有太多的对象了，我怎么才能知道缺失了什么权限？

回答：可以通过如下步骤：

- ➢ 使用 SYSTEM 登录 SQL Editor；
- ➢ 调用语句 call get_Procedure_objects；
- ➢ 将你的存储过程名字输入并执行，这个时候权限检查的结果已写入相应表；
- ➢ 执行语句 SELECT * FROM PROCEDURE_OBJECTS；
- ➢ 检查你是否有相应权限；
- ➢ 执行 TRUNCATE_PROCEDURE_OBJECTS 清除临时数据；
- ➢ 检查是否有分析权限授予了这个用户；
- ➢ 作为测试用，也可以先授予 CONTENT_ADMIN 或者 MODELER 的角色给用户，这些角色已经默认拥有了“看见所有”的权限。



- 当我想选择一个视图时，系统告诉我没有相应权限。但是视图中可能包含有太多的对象了，我怎么才能知道缺失了什么权限？

回答：可以通过如下步骤：

- ➢ 使用 SYSTEM 登录 SQL Editor；
- ➢ 调用语句 call get_accessed_objects_in_statement；
- ➢ 将你的 SQL 选择命令输入并执行；
- ➢ 检查结果将被直接显示；
- ➢ 检查你是否拥有相应的权限。

本章小结与练习

在本章中，我们介绍了用户管理的一些基本概念，如特权、角色与用户等。特权是最基本的授权单位，它可以被赋予角色或用户。角色也可以是有嵌套关系的，即一个角色可以被赋予另外一个角色。然后，我们进一步深入地了解了特权的类型以及一些预定义的角色和用户。

特权包括：

- 系统特权(System Privilege)
- 对象特权(Object Privilege)
- 分析特权(Analytic Privilege)
- Schema 特权(Schema Privilege)
- 程序包特权(Package Privilege)
- 应用特权(Application Privilege)



其中分析特权可以使我们对不同的用户进行不同的行级别的访问数据限制。然后我们对特殊用户_SYS_REPO 的授权进行了详细的分析。

在本章的最后，我们通过权限管理的一些常见任务与问题进一步加强了对本章内容的理解。

练习

1. 创建一个新用户，并通过该用户登录到 SAP HANA 系统。
2. 查看创建的新用户所拥有所有特权。
3. 选择该用户能够访问的某一分析视图，尝试对其进行数据限制。

第七章　SAP HANA 数据供应

第一节　简　介

现在，让我们进行正式接触 SAP HANA 之前的另一准备工作——数据供应。所谓“工欲善其事，必先利其器”，相信很多开发人员都有这样的体会，事先准备好的优秀数据往往可以使你在开发过程中目标明确，起到事半功倍的效果。对 SAP HANA 而言，相比较于市场上已有的传统数据库(如微软的 SQL SERVER 等)，SAP HANA 自身对于各种数据类型的支持并不是特别完善，往往需要其他 SAP 的工具协同来完成工作。 现在已有的向 SAP HANA 中导入数据的方法有以下几种：

- 本地文本文件导入
  - 是原生 HANA 功能，不需要使用其他工具；
  - 用于演示或导入少量数据。
- CTL 方法
  - 是原生 HANA 功能，不需要使用其他工具；
  - 需要一个.ctl 文件来控制；
  - 数据源：可以导入 CSV 文件；
  - 性能优异，用于导入大量数据。
- 基于日志的复制(Sybase Replication)
  - 数据源主流关系数据库；
  - 可实时进行数据复制。
- 基于 ETL 的复制(Data Service)
  - 数据源：可以是各种类型(甚至是无结构的，如 Facebook、微博数据等)。
- 基于触发的复制(SLT Replication)
  - 源数据：可以是 SAP ERP 或主流关系数据库；

➢ 可实时进行数据复制。

- 提取器直接抽取(Direct Extractor Connection)
  ➢ 数据源：只能是 SAP ERP 系统，且需要安装 SAP BI CONT。

在这些方法之中，本地文本文件导入与 CTL 方法相对较简单，可在数据源较为简单、需要一次性导入时使用。我们可以从图 7-1 中可以看到针对不同的数据源，目前 SAP 所支持与推荐的不同方法。

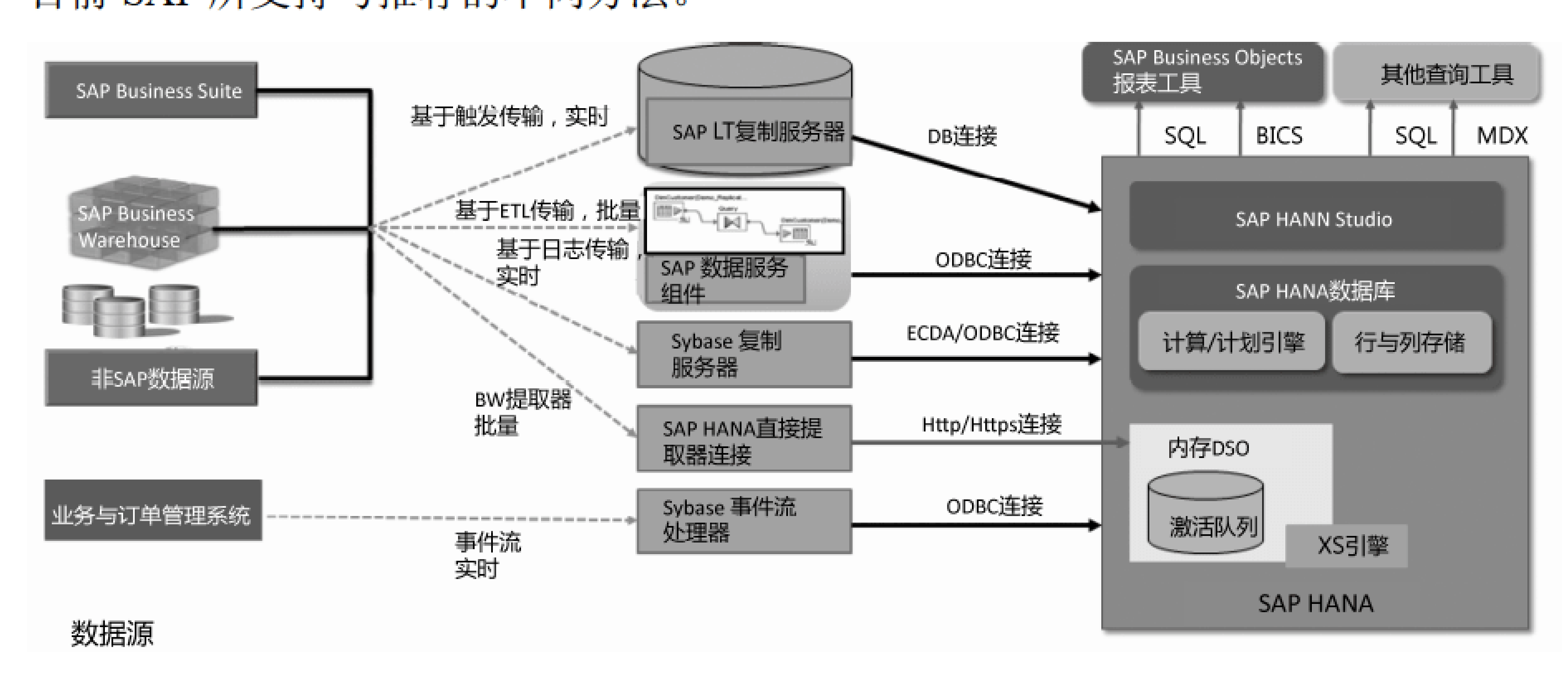


图 7-1

这几种方式各有优劣，可基于企业的实际情况来选用。在本章的接下去几节，我们会从最简单的文本文件开始，然后对最流行的(SLT)与功能最强大的(ETL)给出详细步骤，以便用户能掌握其实际应用。

第二节　本地文件方式数据导入

在对 SAP HANA 进行各种操作之前，我们所面对的不可避免的第一步就是数据的导入导出。在这里，我们可以将 SAP HANA 看做是一个普通的关系型数据库。作为开发者，将测试数据导入 SAP HANA 最简单的办法就是利用 SAP HANA Studio，从文本文件或者 CSV 文件导入。在本节，我们将此作为 SAP HANA 数据导入的第一步来演示：

(1) 找到 DATA.ZIP 并解压缩。

(2) 打开 SAP HANA Studio，进入“Qucdik Launch”快速启动界面。如果未显

示，可打开菜单“Help”→“Quick Launch”(见图 7-2)。

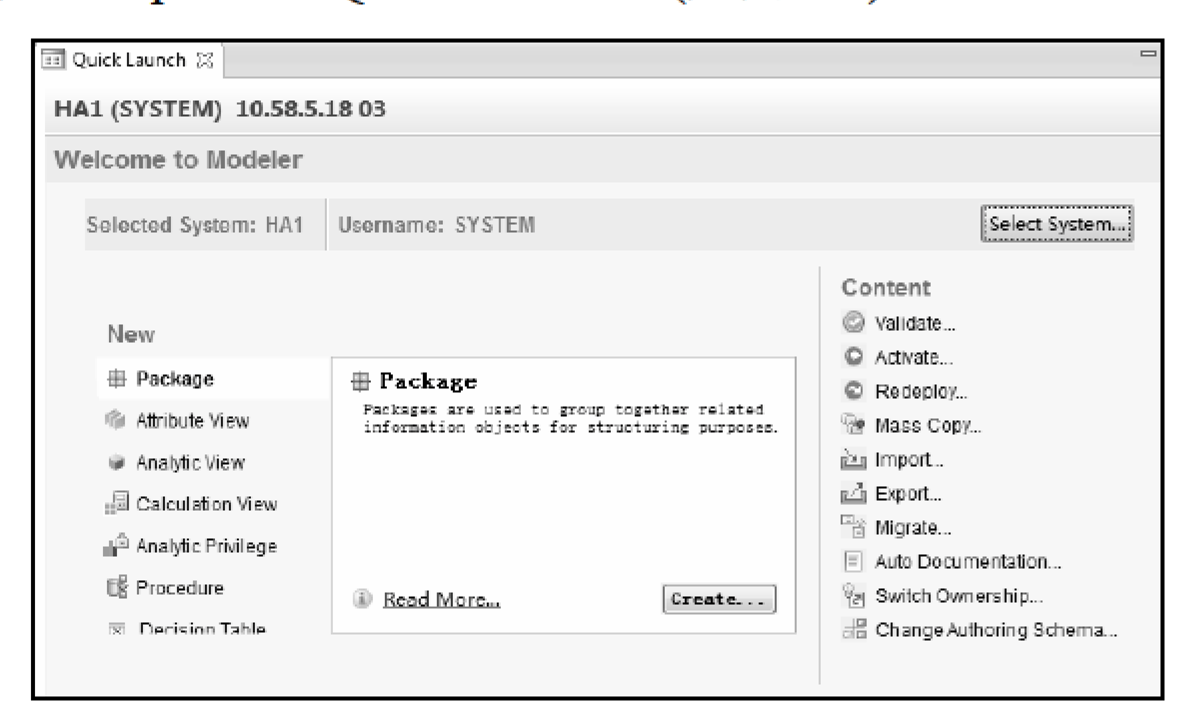


图 7-2

(3) 单击“Select System”按钮选中需要导入数据的系统(见图 7-3)。

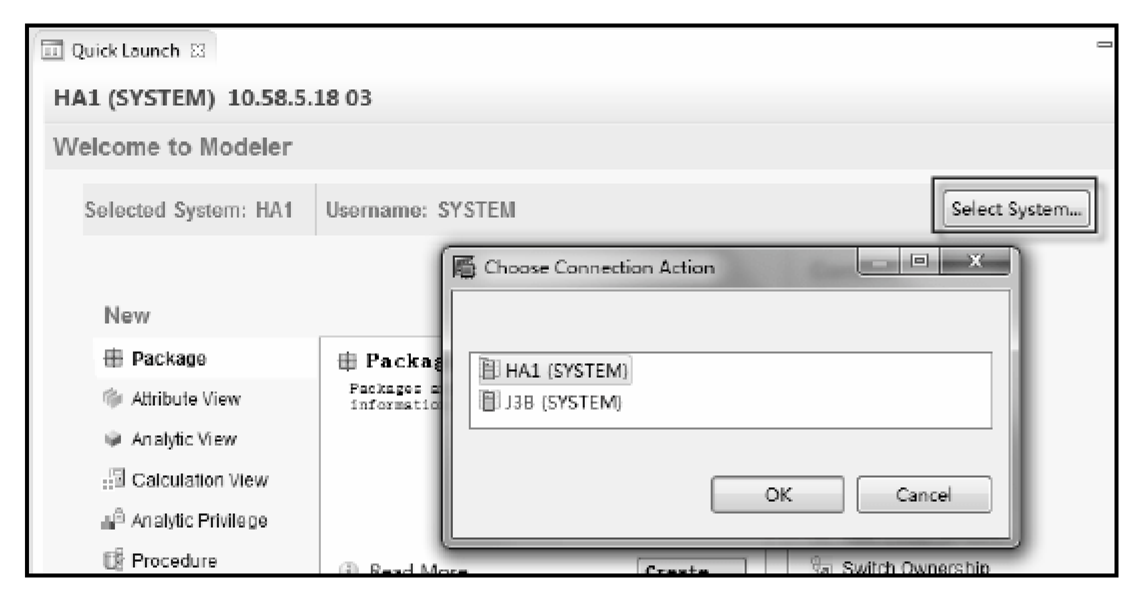


图 7-3

(4) 单击“Import”选项(见图 7-4)。

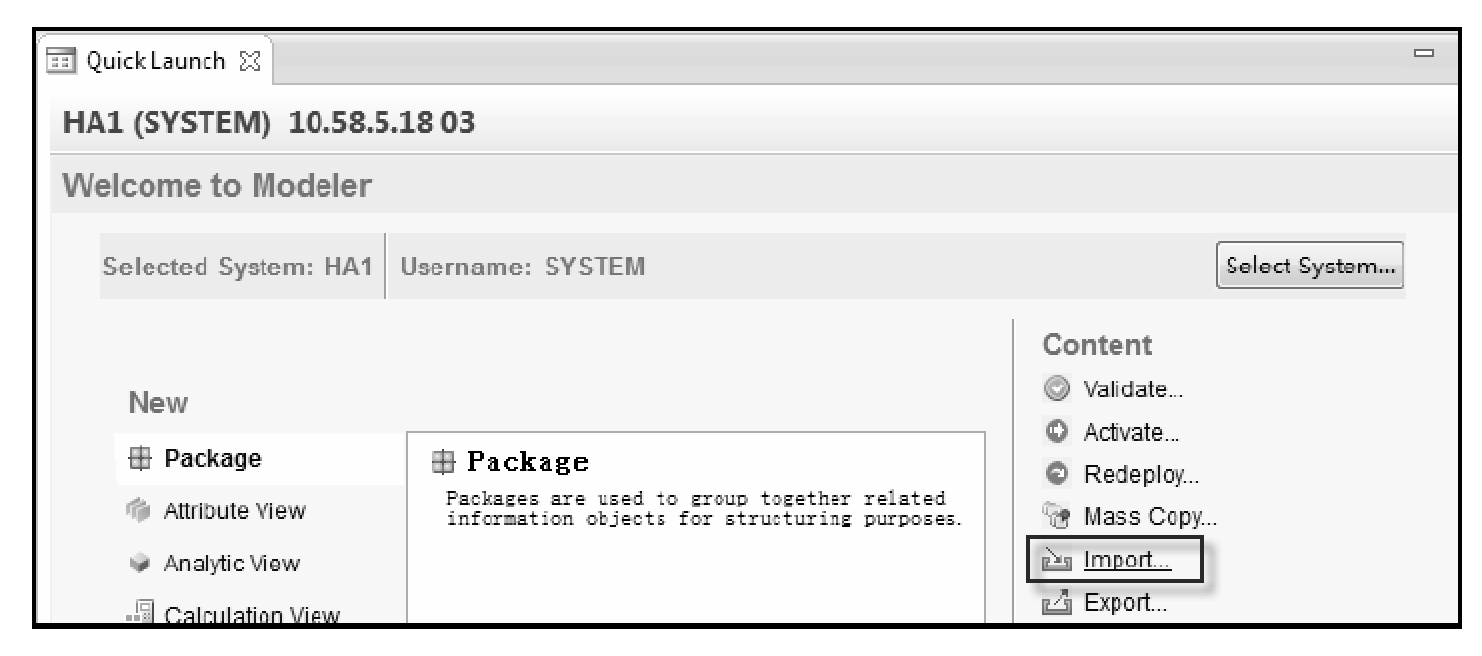


图 7-4



(5) 在新出现的界面中，展开“SAP HANA Content”节点，选择“Data from Local File”选项，然后单击“Next”按钮(见图 7-5)。

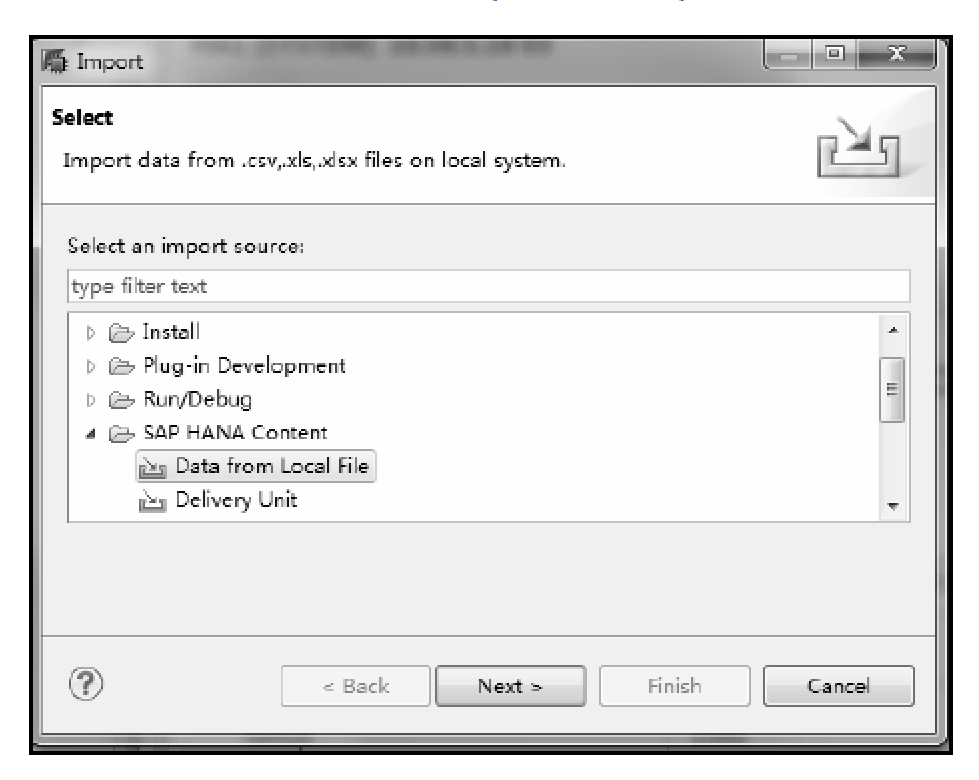


图 7-5

(6) 参照图 7-6 配置上传文件的参数。

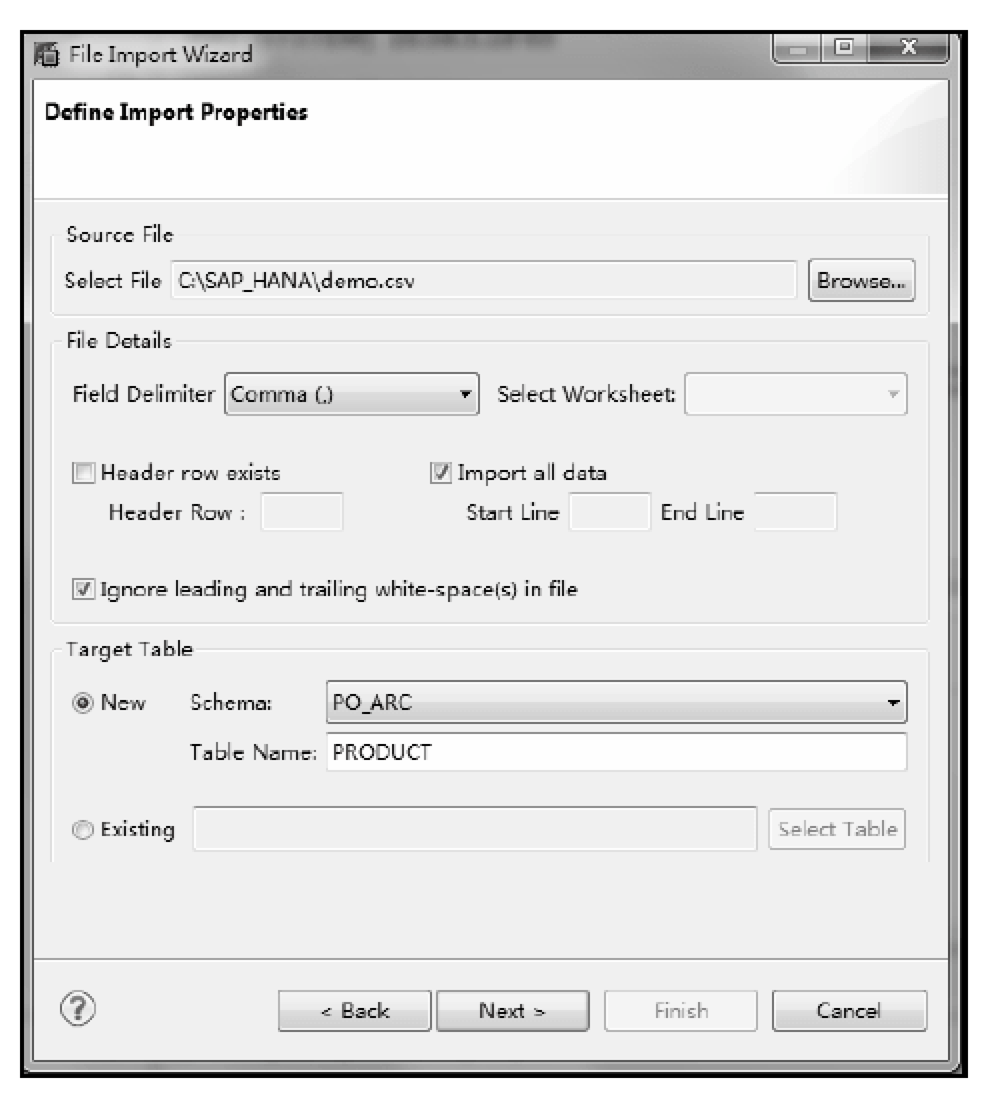


图 7-6



从图 7-6 中我们可以看出，在“Source File”托盘中我们能选择需要上传的文件；在“File Details”托盘中我们能选择源数据的分隔符，如果你上传的是 Excel 格

式的文件，你还能在此处选择需要上传的工作表。随后你可以设置上传数据是否含有表头以及是否上传全部数据，并且设置对于源数据中所包含的起始及结尾的空格如何处理(包含或者忽略)；接下来在“Target Table”托盘里面，你能设定要上传的本地文件是否需要新建 Schema 和数据表与之对应，或者是要上传到已经存在的数据表中。

一切就绪后，单击“Next”，SAP HANA Studio 会对需要上传的数据自动进行检测然后触发上传数据操作。到这里我们可以看出，从本地文件导入，这是最简单的方法，但往往也是最不灵活的，并且有限制性。例如，如果数据文件有缺陷，那么从 SAP HANA Studio 导入时会失败。数据的提交是对整个文件来说的，因此其错误日志没有什么意义。所以，如果它可以工作，那么就简单好用。否则，它几乎不会告诉你哪里错了。而且，这种方法最适合较小的文件。但是，你也可以导入保存在 HANA DB 服务器(或者通过 NFS 或其他方式加载到 HANA DB 服务器上的远程硬盘)上的数据。《SAP HANA SQL 参考指南》中有关于 SQL 语句 IMPORT FROM 的最新文档，需要的话可进行参考。

第三节　CTL 方式数据导入

在上节中，我们讲述了直接使用 SAP HANA Studio 进行本地文件导入。该方法简单直接，但是当 CSV 文件数据量特别大时，我们可以考虑使用 CTL 方式来进行导入。同样，接下来我们通过一个实例来了解如何通过 CTL 方式导入平面数据。

(1) 在开始之前，先确认你需要导入的 Schema 名字与服务器上存放文件的地址。如 Schema 名为 PO_ARC，文件地址为/usr/sap/<HANA instance>/HDB00/work/。

(2) 准备好 CSV 文件，需要注意的是在 CSV 中没有列名信息。

(3) 在 SAP HANA 中创建相应的表结构，具体操作请参考第八章第二节。

(4) 创建一个控制文件，参考语法如下：

```
[IMPORT DATA]
INTO TABLE <schema>.<tablename> [(<field specs>)]
FROM <os_data_filename>
[RECORD DELIMITED BY '<eor>']
[FIELDS [DELIMITED BY '<eoc>']
[OPTIONALLY ENCLOSED BY '<moq>']]
```

```
[ERROR LOG <os_bad_filename>]
```

在本例子中，该文件示例内容如下：

```
IMPORT DATA
INTO TABLE PO_ARC.USERS
FROM ' /usr/sap/<HANA instance>/HDB00/work/USERS.CSV'
ERROR LOG ' /usr/sap/<HANA instance>/HDB00/work/USERS.ERR'
```

将文件存为 USER.CTL。

(5) 将 USER.CTL 与 USER.CSV 上传至文件夹/usr/sap/<HANA Instance>/HDB00/work/。

(6) 执行 IMPORT，参考语法如下:

```
IMPORT FROM <control_file>
WITH [THREADS <thread_num>] [BATCH <batch_size>]
WITH TABLE LOCK [WITHOUT TYPE CHECK]
```

在本示例中，我们可以使用命令：

```
IMPORT FROM '/usr/sap/HA1/HDB03/work/USER.CTL'
```



当遇到错误时，请仔细检查文件在 LINUX 下的访问权限及字母大小写。

(7) 在 SAP HANA Studio 中，找到在第三步创建的数据表，并右击“Open Content”打开，检查数据表内容，确认所有数据已导入。

第四节　SLT 方式数据导入

谈到 SAP HANA 的数据导入，就不得不谈到 SLT 这种方式。 SLT 的全称是 SAP Landscape Transformation，在 SAP HANA 出现以前，它是基于 SLO 的技术。在过去的 10 年中，其每年都被数以百计的 SAP 相关实施项目使用，是一种已被长时间验证的可靠技术。据统计，现在大约 85%的 SAP HANA 客户都会使用 SLT 技术来实现数据导入。关于 SAP HANA 相关的 SLT 组件的最新安装及讨论，可以参考如下链接：

- http://help.sap.com/hana_appliance/
- https://community.wdf.sap.corp/sbs/groups/lt-replication-for-hana?view=documents

在前面的小节中，我们已经学会了如何将数据从文本文件导入 SAP HANA 系统。在前面两个例子中，我们一直是将 SAP HANA 完全当做一个关系型数据库。但我们知道，SAP 是全世界最大的 ERP 系统供应商，因此在我们讨论 SAP HANA 时，可以想象很多时候 SAP HANA 的用户并不是纯粹将 SAP HANA 当做数据库使用的。他们更可能会考虑的是，如何将 SAP HANA 与他们已有的企业 ERP 系统相结合，而 SLT 正是实现这一结合的首选方式。

一、SLT 功能

由于 SLT 是基于 SAP 的 Netweaver 平台，因此一台独立的 Netweaver 服务器是安装 SLT 的必要前提条件(也可以重用原有 ERP 系统)。但是我们需要再一次强调，SLT 不仅可以抽取 SAP 系统的源数据，同时也支持第三方数据库源数据的抽取。图 7-7、图 7-8 是 SLT 的结构图。

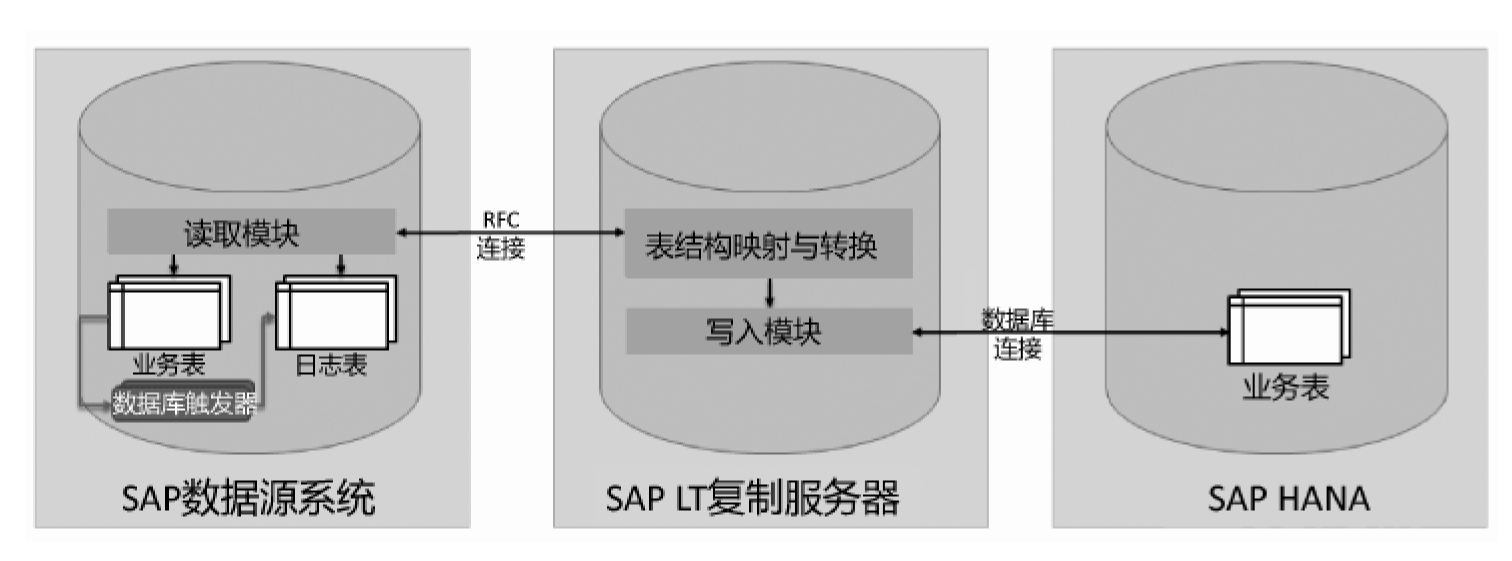


图 7-7

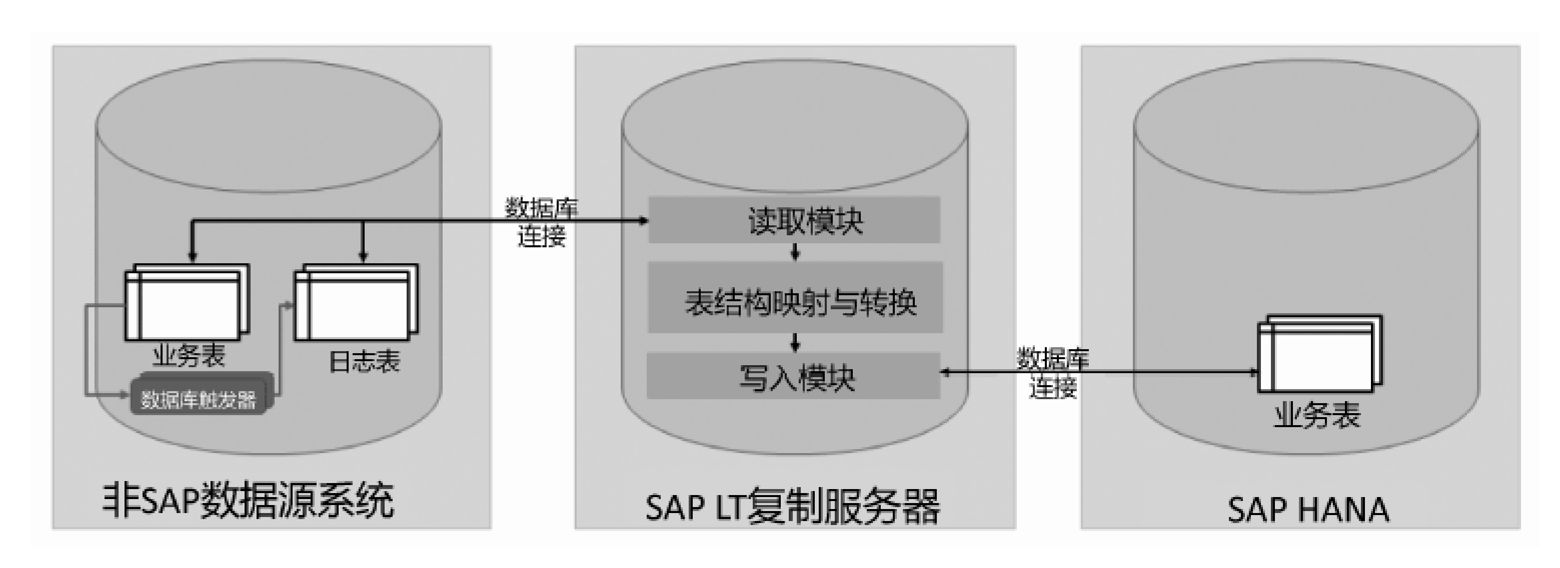


图 7-8

使用 SLT 的好处：

- 可以实时复制相关数据导入 SAP HANA 系统；
- 在复制数据时可以转换成 HANA 格式；
- 使用已证明的 SLO 技术(几乎 0 关机时间)；
- 迅速简单实现且完全与 SAP HANA 界面集成；
- 几乎所有 SAP 系统都支持(SAP R/3 4.6 以后)；
- 支持非 SAP 的源系统。

二、操作步骤

SLT 复制方式的建立需要三个步骤：准备工作，建立 Schema 与复制表。在这里，我们以 SLT 直接复制数据库表作为示例，一步一步讲解具体步骤。由于 SLT 是基于 SAP Netweaver 的，这里用户需要一个 SAP 的系统环境且对 SAP 的操作有一定了解。

1. 准备工作

(1) 在 SAP 系统中，输入事务代码 DBCO 并运行(见图 7-9)。

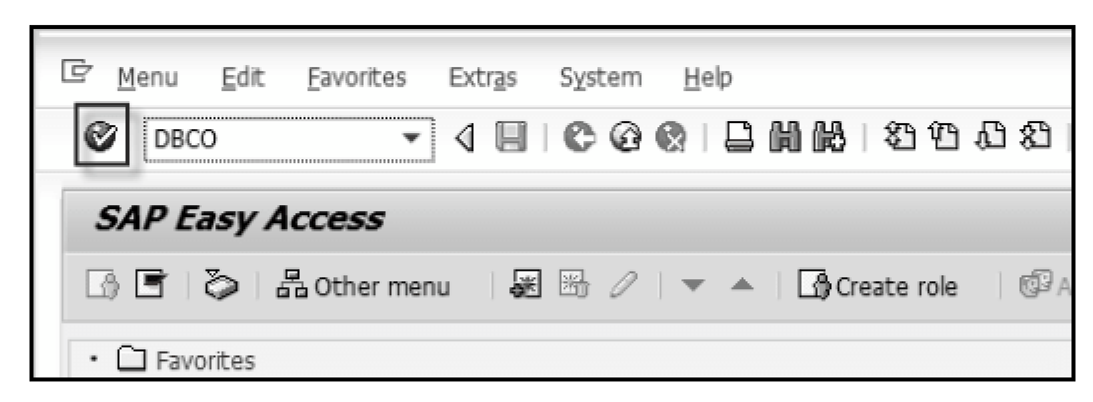


图 7-9

(2) 选择”New Entries”(见图 7-10)。

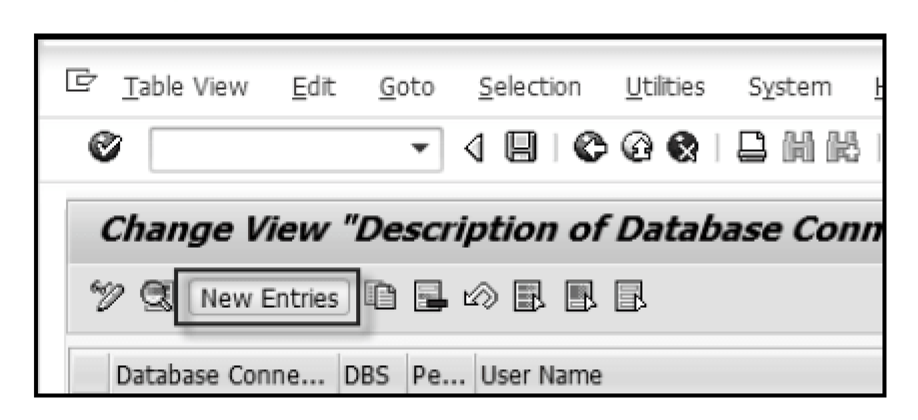


图 7-10

(3) 在新出现的窗口中，输入如表 7-1 所示的信息，界面如图 7-11 所示。



表 7-1

字　　段	解　　释
DB Connection	连接名
DBMS	源数据库种类，如 Sybase ASE
User Name/DB password	用户名/密码
Conn. Info	连接信息

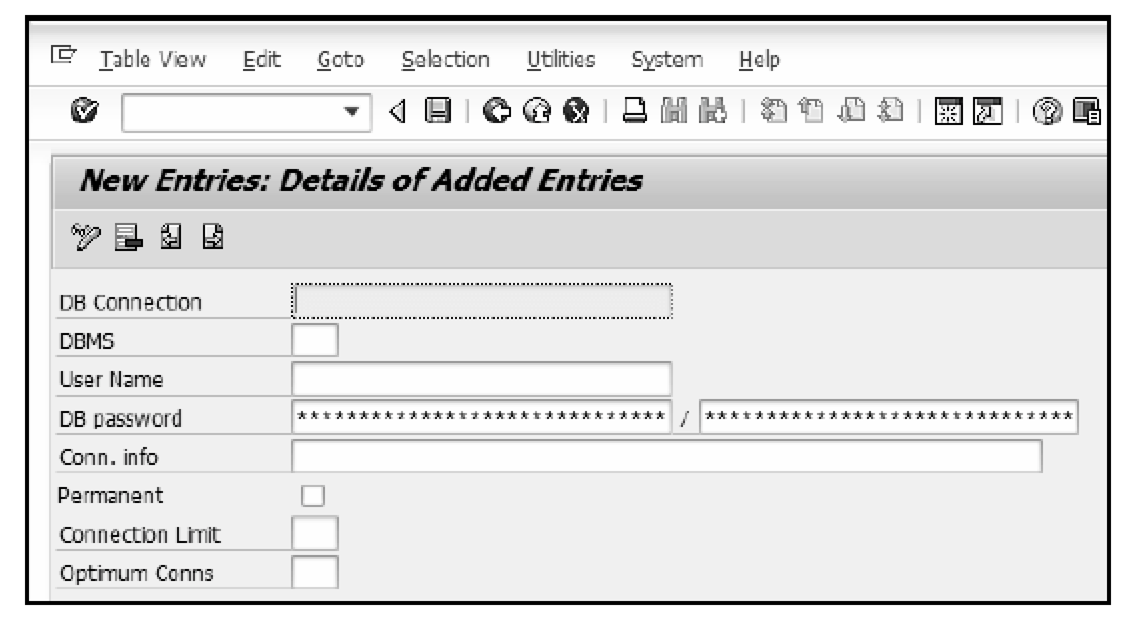


图 7-11

(4) 单击保存按钮(见图 7-12)。



图 7-12

(5) 可以输入事务代码“SA38”并输入程序名“ADBC_TEST_CONNECTION”来对连接进行测试(见图 7-13)。

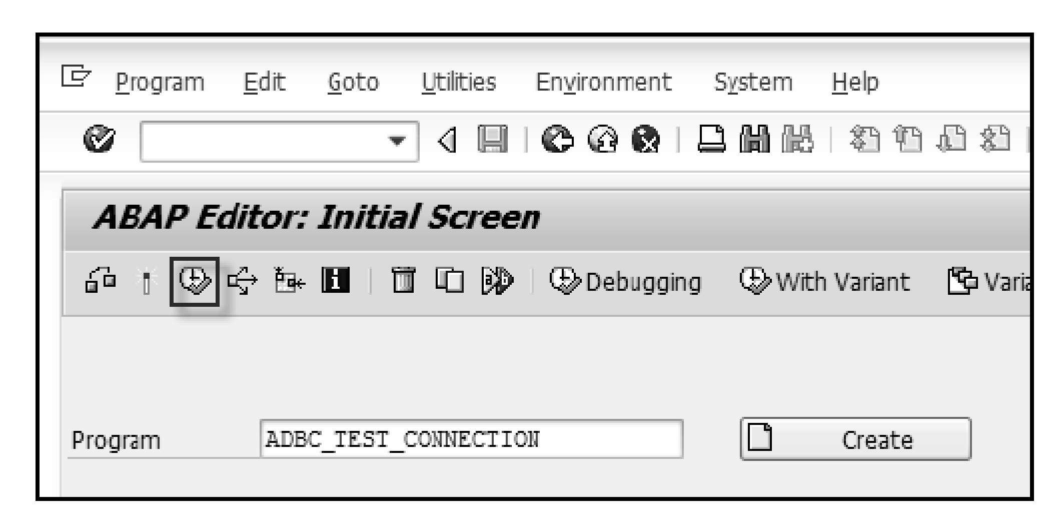
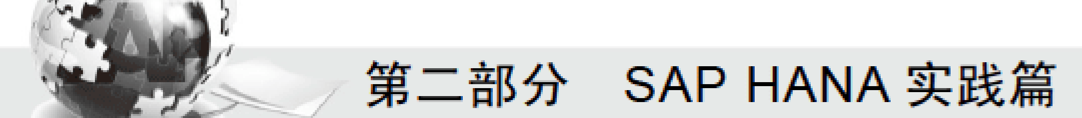
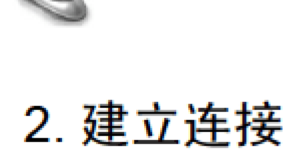


图 7-13





2. 建立连接

(1) 在 SAP 系统中，输入事务代码 LTR 并运行(见图 7-14、图 7-15)。

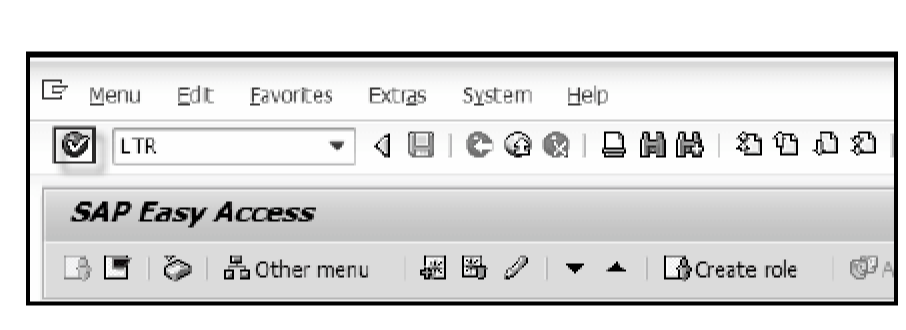


图 7-14

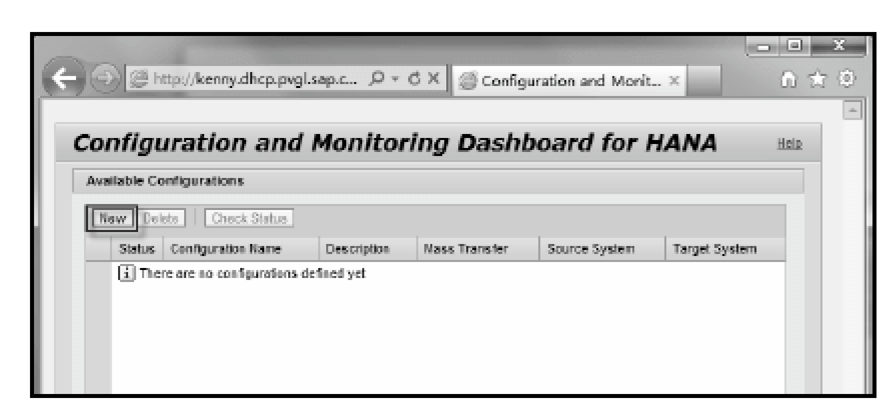


图 7-15

(3) 在出现的对话框中，输入如表 7-2 所示的信息。

表 7-2

字　段	解　释
Configuration Name	连接名，会在 HANA 中创建同名 Schema
No. of Data Transfer Jobs.	在 SLT 系统中的任务数
Connection to Source System	源数据库种类，这里以 SQL Server 示例
User Name/DB password	用户名密码
Conn. Info	连接信息，这里以 SQL Server 为例

图 7-16 为已填写好的例子，以供参考。

Create a New Configuration

General Data
Configuration Name: * HANA_SLT
Description:
Application:
No. of Data Transfer Jobs: * 003

Connection to Source System
SAP System　Non-SAP System
Database System: * MSSQL
DB Connection: * PO_ARC
Schema Name: * dbo

Connection to HANA System
System User Name: * SYSTEM
Password: * ●●●●●●●●●
Host Name: * 10.58.5.18
Instance Number: * 03

Tablespace Assignment
Data Class:

Replication Options
Real Time
Schedule by Interval 000
Schedule by Time 00:00:00

OK　Cancel

图 7-16

(4) 输入好信息后，单击“OK”按钮。在你的 SAP HANA Studio 客户端中，你



应该能找到和你输入的“Configuration Name”名称一致的 Schema。

3. 复制表

(1) 打开 SAP HANA Studio 并调出快速启动透视图，在“DATA”区域中单击“Data Provisioning”(见图 7-17)。

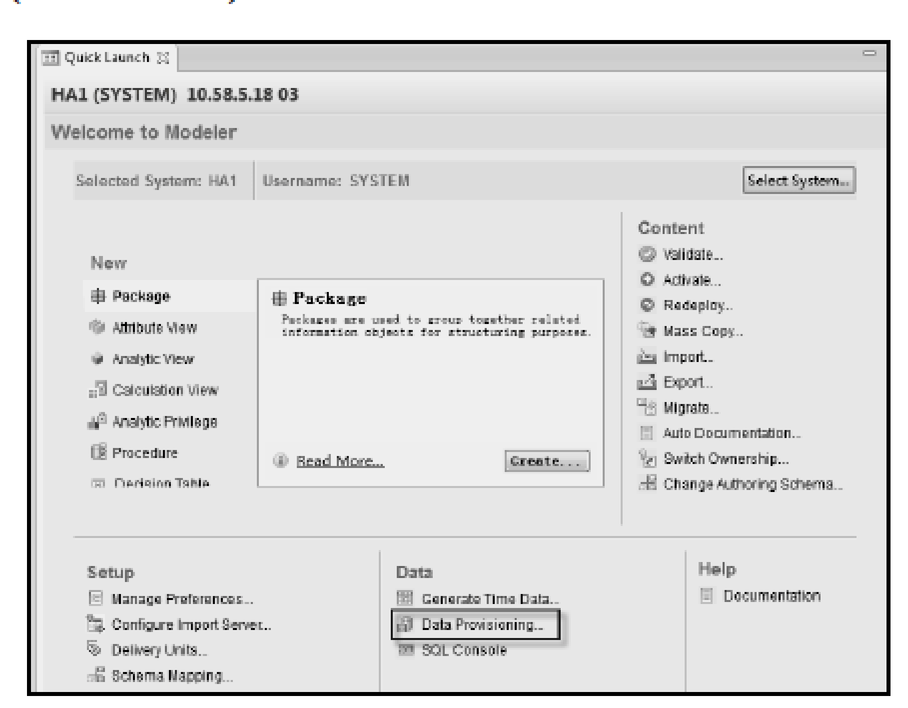


图 7-17

(2) 选择源系统和 Schema(见图 7-18)。

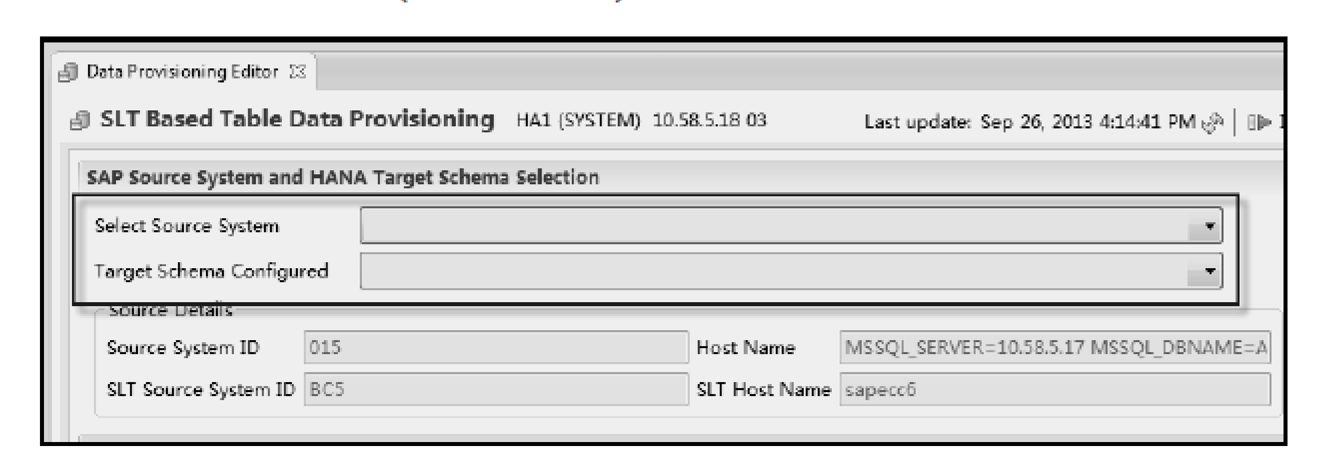


图 7-18

(3) 单击“Replicate…”按钮(见图 7-19)。

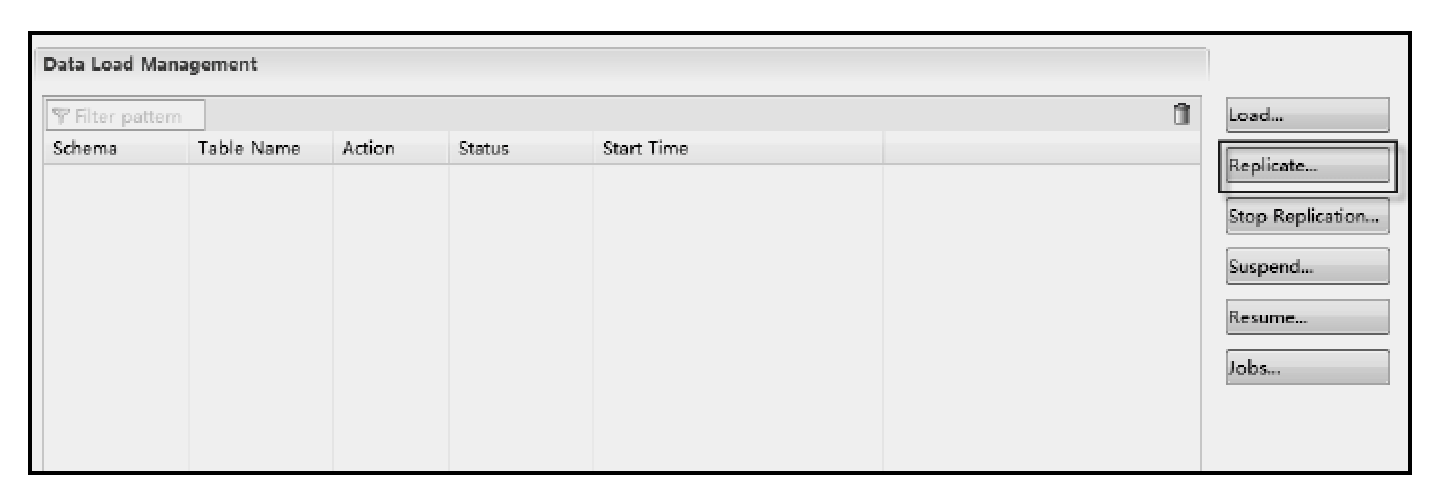


图 7-19



(4) 选中要添加的表并添加至右边的选中框(见图 7-20)。

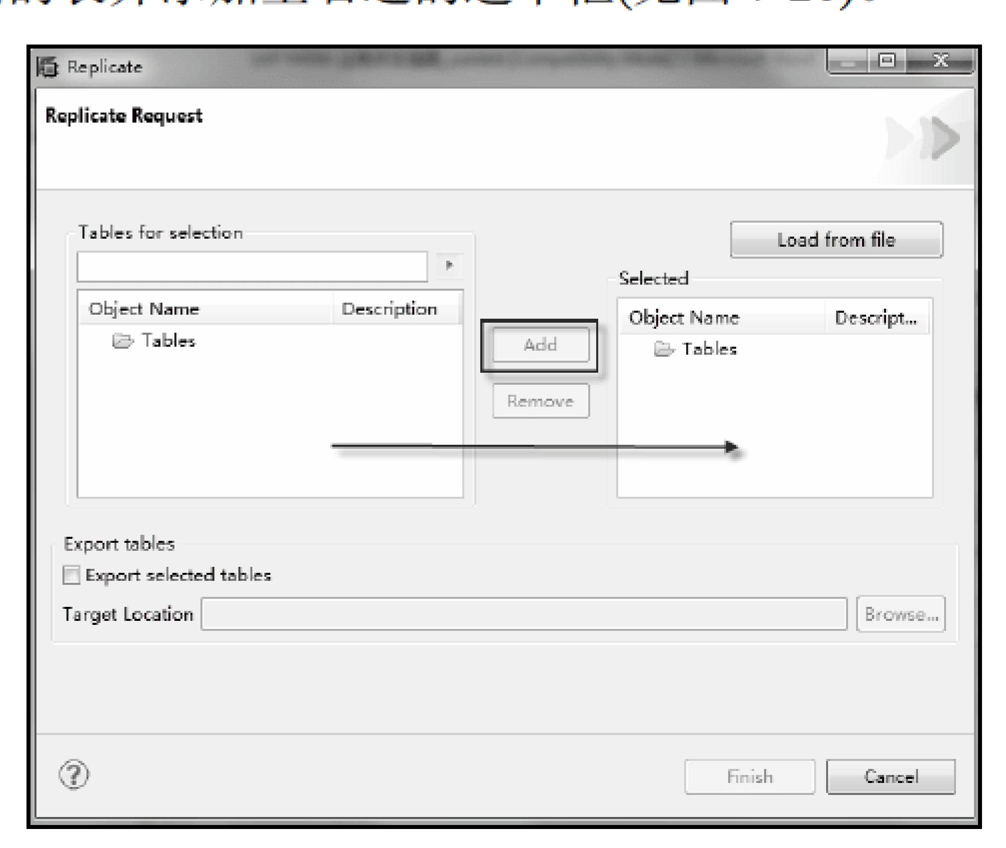


图 7-20

(5) 单击“Finish”按钮，复制表的进程就会在“Data Load Management”框中出现。

(6) 当所有表的状态都变为“Replicate In Process”时，表示所有数据复制都已完成。

(7) 我们可以在 SAP HANA Studio 中通过 SQL 语句来检查复制表的内容是否完整。



第五节　ETL 方式数据导入

相信大家对 ETL 这个名词都很熟悉，它是英文 Extracting(抽取)、Transform(转换)和 Load(载入)的首字母缩写。ETL 在传统数据仓库中具有相当重要的地位。在这里我们将以 SAP 的 ETL 工具“Data Service”为例，来学习如何通过 ETL 的方式将数据导入。

当实时性不是最重要的指标的时候，SAP 数据服务(SAP Data Service)是首选的数据复制的方式。SAP 数据服务拥有以下特点：

- 高性能
  - 利用高度可伸缩(scalable)引擎来复制大量数据至 SAP HANA；

  - 可与 SAP HANA 的 bulk-load 接口集成。
- 广泛的连接性
  - 可以连接所有主要的企业源数据类型；
  - 可以快速连接至企业应用、数据库文本文件等。
- 强大的转换功能
  - 可任意对数据进行清洗转换；
  - 强大的内置数据质量检查；
  - 支持费关系型数据，如文本与 XML 等。

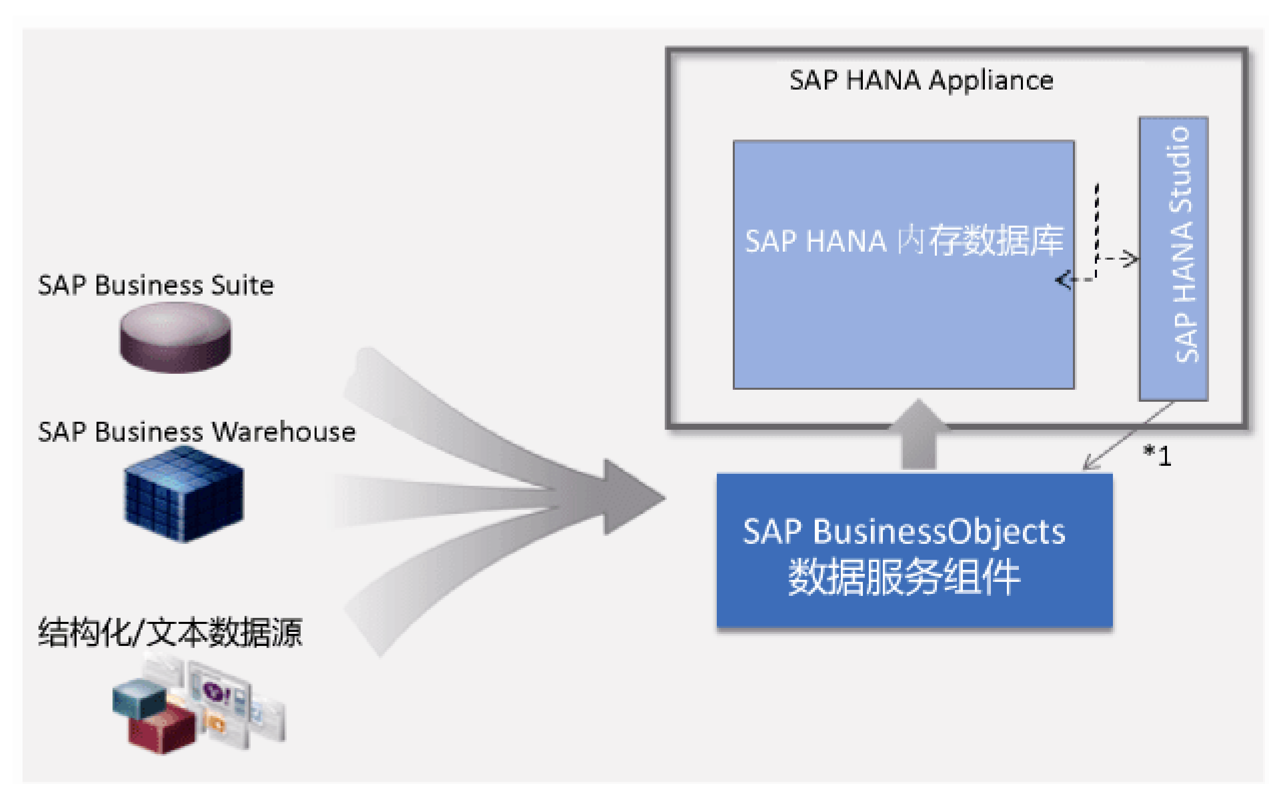


图 7-21

一、SAP 数据服务简介

在开始数据服务组件复制数据之前，我们先了解一下数据服务组件的一些主要组件与概念，如表 7-3 所示。

表 7-3

组件类型	组　　件	说　　明
服务器组件	Job Server	执行数据服务成批任务
	Access Server	执行数据服务实时任务
客户端组件	Designer	开发客户端
	Repository	用于储存所有数据服务相关的元信息(如数据流等)



(续表)

组件类型	组　件	说　明
客户端组件	Data Store	分为源数据存储与目标数据存储，通过数据服务的任务完成不同数据存储之间的数据传输。
	Management Console	网页应用程序用于管理数据服务。

在我们进行下面的操作之前，请确保服务器与客户端组件都已准备完成。目前 SAP 数据服务与 SAP HANA 的集成并不是特别紧密，仅限于从 SAP 数据服务将元数据(metadata)导入 SAP HANA。在本书中，我们将 SAP HANA 看做是一个普通的数据库，然后讲述如何通过 SAP 数据服务复制数据至 SAP HANA 数据库。这可以分为以下三个步骤：

- 建立 SAP 数据服务仓库；
- 建立 SAP HANA ODBC(作为数据存储准备)；
- 建立 SAP 数据服务任务实现数据传输。

二、使用 SAP 数据服务进行数据传输

我们在上一小节介绍了使用 SAP 数据服务进行数据传输的三个步骤，在本小节中我们看看如何来具体操作这些步骤。

1. 建立 SAP 数据服务数据仓库

(1) 在 SAP 数据服务组件的操作系统中，打开开始菜单，找到“SAP BusinessObjects Data Service”项。

(2) 打开“Data Service Repository Manager”菜单。

(3) 在出现的对话框中，输入如表 7-4 所示的信息，如图 7-22 所示。



表 7-4

字　段	解　释
Repository type	选择“Local”
Database type	选择 SAP 数据服务仓库的数据库类型
Database server name	输入数据库服务器名称
Database name	输入数据库名称
User name	输入用户名
Password	输入密码

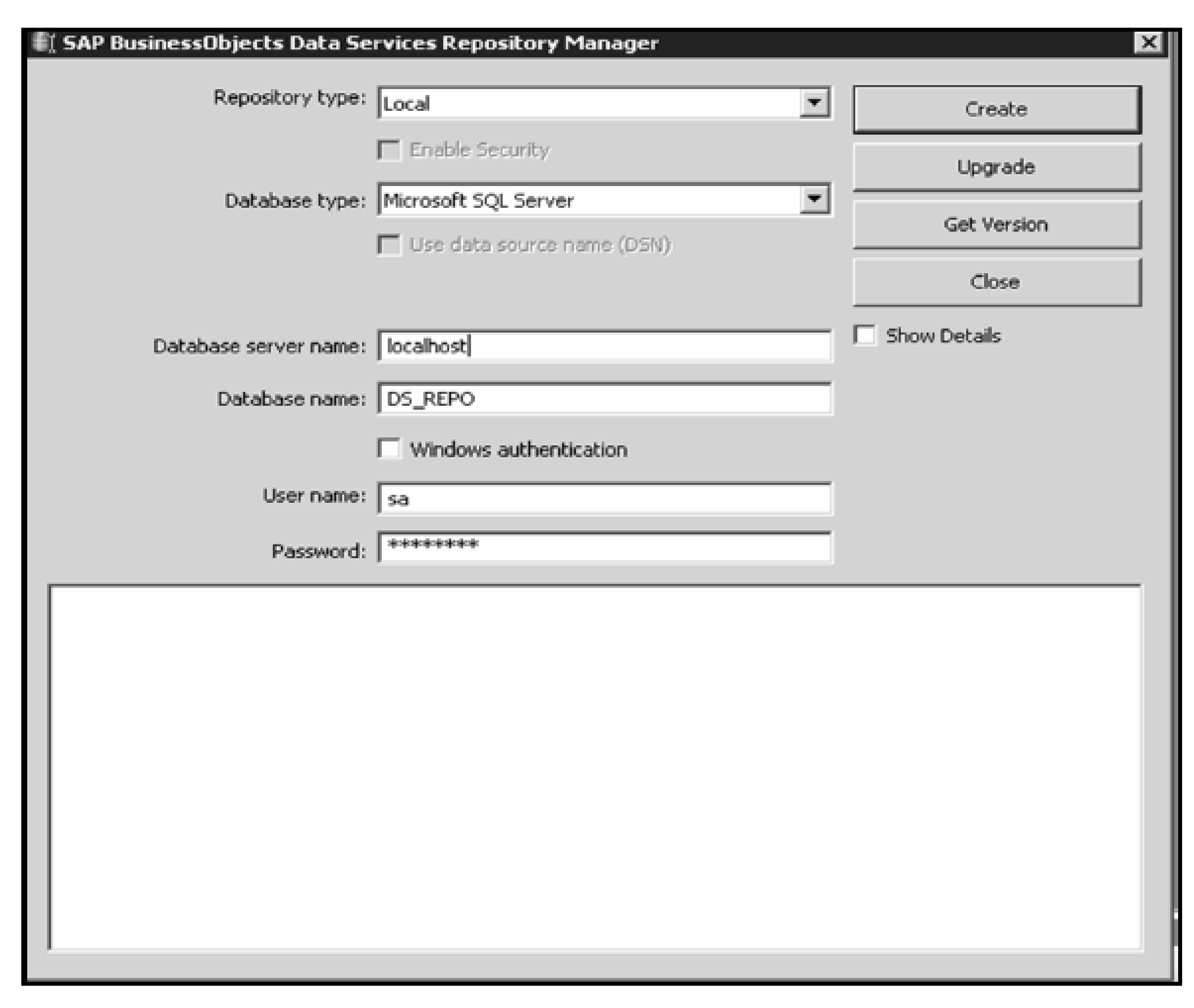


图 7-22

(4) 单击创建“Create”按钮，可以看见数据库创建成功信息(见图 7-23)。

SAP BusinessObjects Data Services Repository Manager
Repository type: Local
Enable Security
Database type: Microsoft SQL Server
Use data source name (DSN)
Database server name: localhost
Database name: DS_REPO
Windows authentication
User name: sa
Password: ********
Create
Upgrade
Get Version
Close
Show Details

Addng Match transform... (BODI-300129)
Addng Associate transform... (BODI-300130)
Addng USA Regulatory Address Cleanse transform... (BODI-300131)
Addng User Defined transform... (BODI-300132)
Addng Global Geocoder transform... (BODI-300189)
Addng Extraction transform... (BODI-300212)
Addng DSF2WS transform... (BODI-300226)
Installing the JIT viewdata job... (BODI-300230)
The local repository was successfully created. (BODI-300055)

图 7-23

(5) 在服务器中输入网址：http://<BOE Server>:8080/BOE/CMC。

(6) 输入登录信息登录后，选择“Data Service”下拉值(见图 7-24)。

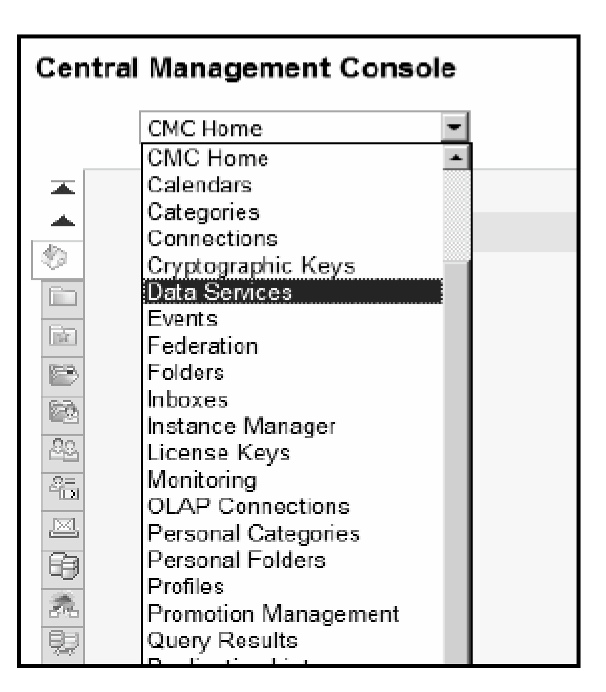


图 7-24

(7) 在“Data Service”管理界面中，右击菜单，选择“Manager”→“Configure repository”(见图 7-25)。

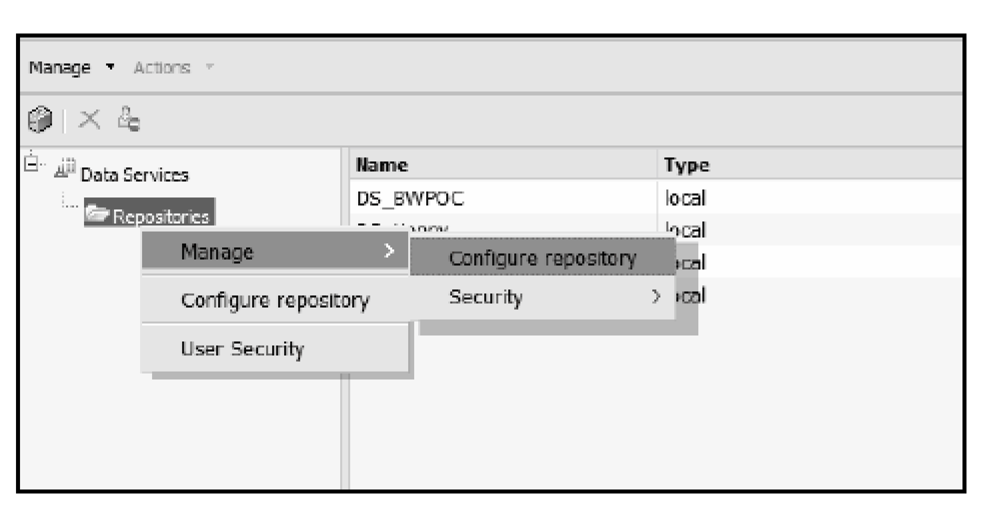


图 7-25



(8) 输入相应信息(见图 7-26)。

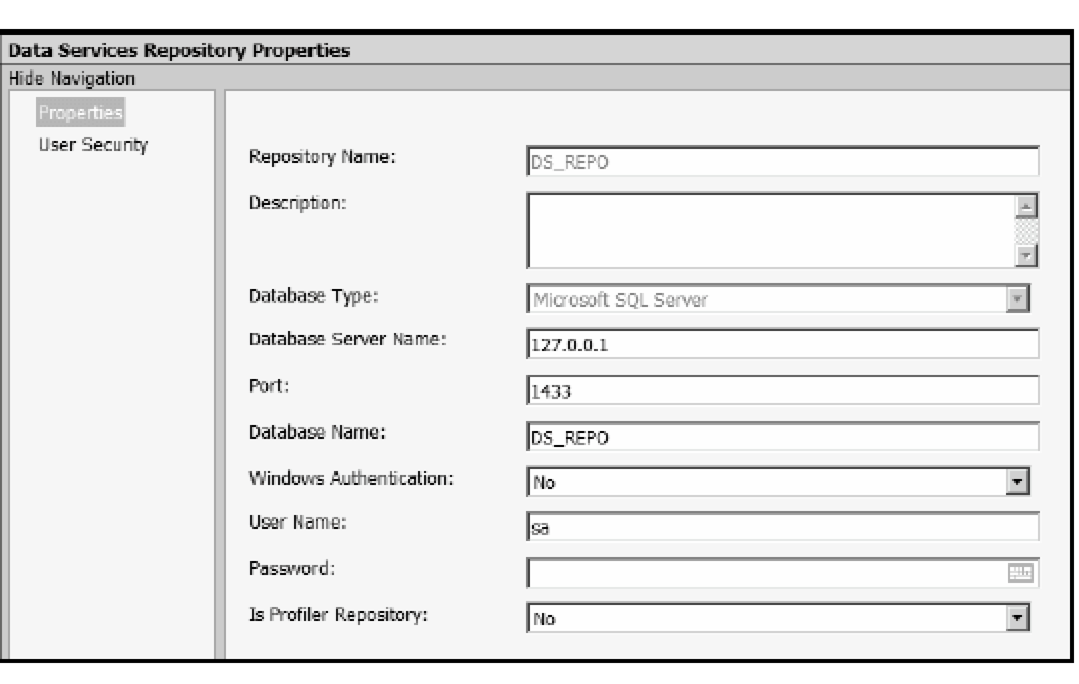


图 7-26

(9) 测试连接并保存，Repository 建立成功。

2. 建立 ODBC

(1) 在 SAP 数据服务的操作系统中，进入开始菜单→管理工具→ODBC 数据源。

(2) 选择系统 DSN，并添加。

(3) 选中 HDBODBC 数据源，单击“Finish”按钮(见图 7-27)。

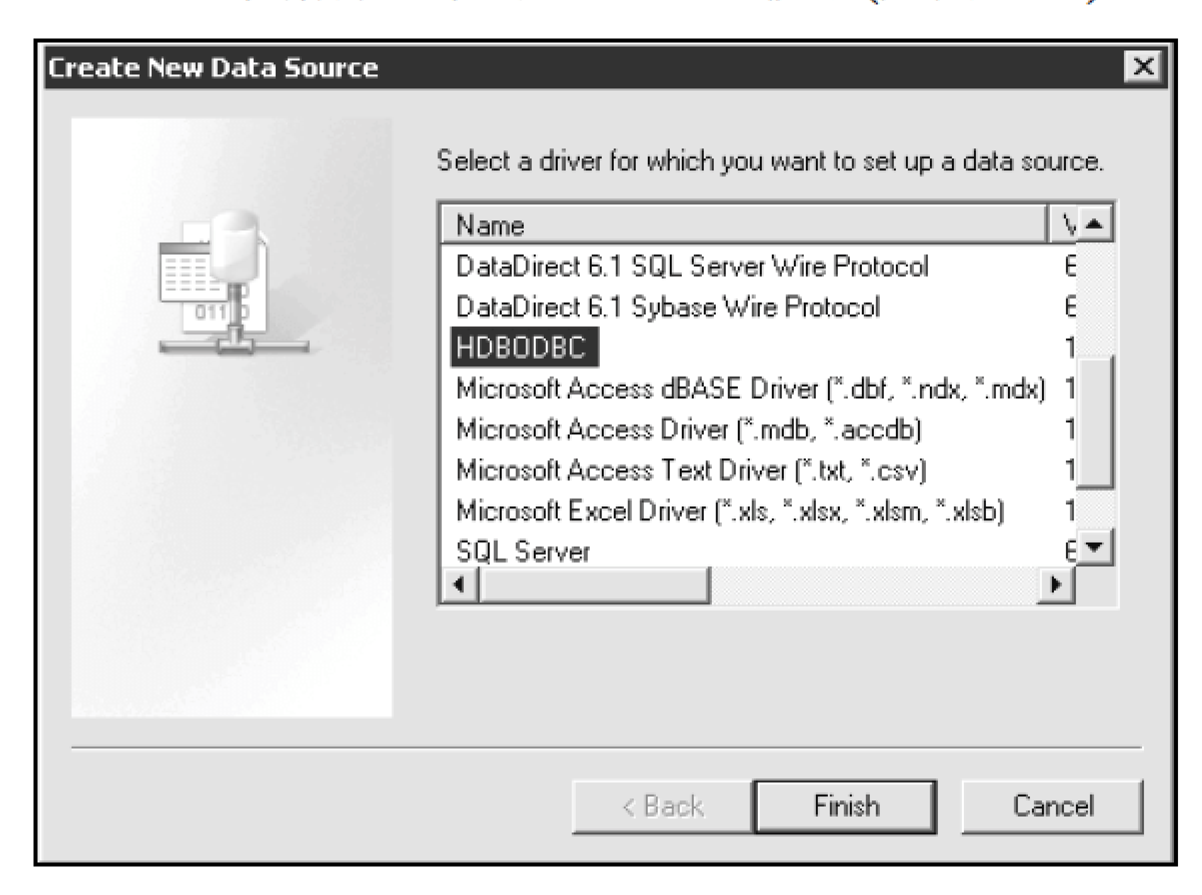


图 7-27

(4) 输入数据源名称与服务器地址，注意端口命名为 3XX15(XX 为 SAP HANA 的 Instance Number)，如图 7-28 所示。

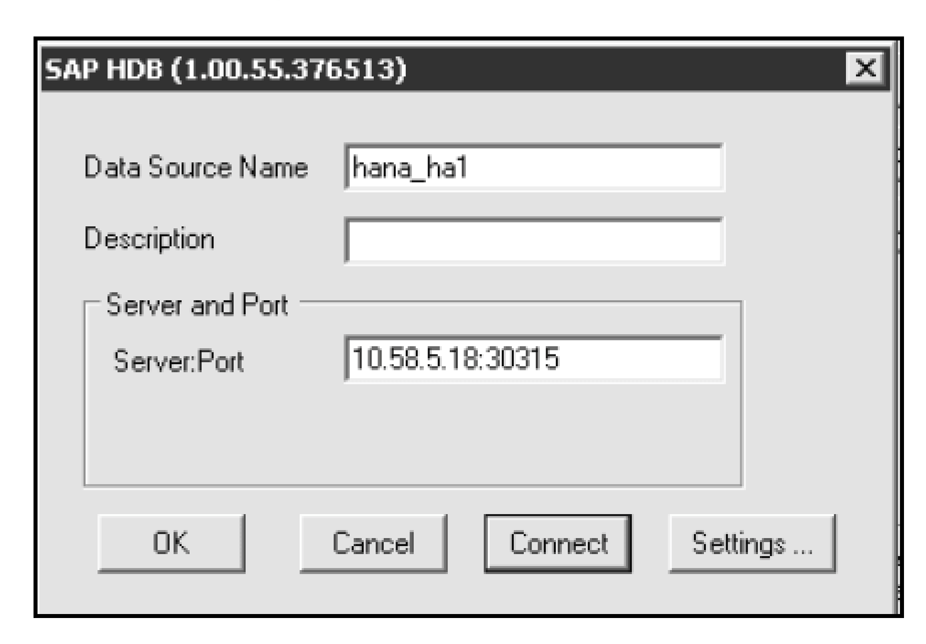


图 7-28

(5) 可以选择“Connect”来测试 ODBC 数据源。

3. 建立 SAP 数据服务任务实现数据传输

(1) 在 SAP 数据服务的操作系统中，开始菜单→找到“SAP BusinessObjects Data Service”。

(2) 打开“Data Service Designer”。

(3) 在出现的对话框中，输入如表 7-5 所示的信息登录 BO 智能平台。

表 7-5

字　段	解　释
System	选择“Localhost”
Username	输入用户名
Password	输入密码
Authentication	选择“Enterprise”

(4) 点击登录后，双击需要选择的仓库(见图 7-29)。

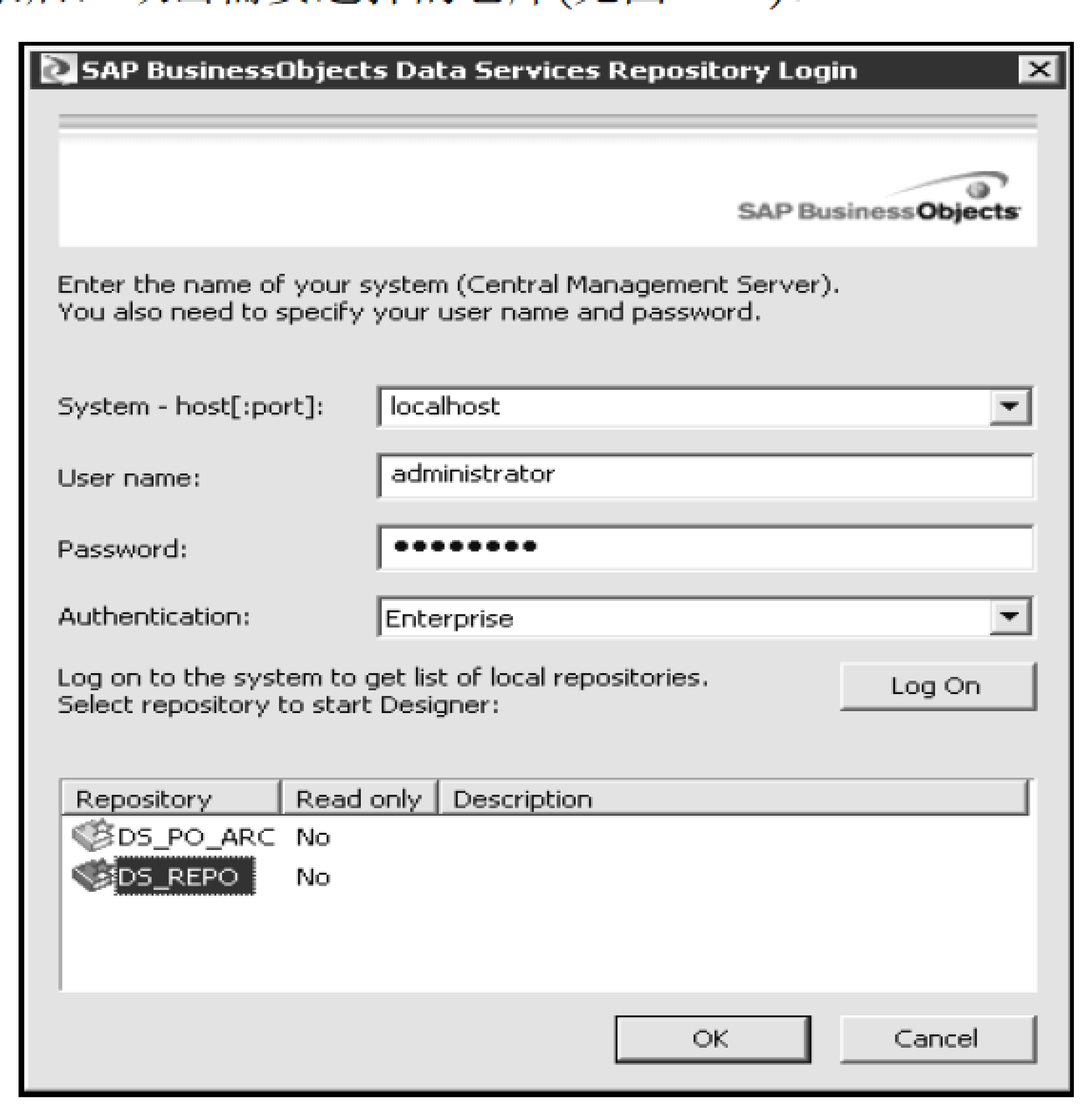


图 7-29

(5) 在出现的界面左下方，可以看见很多可选标签。

(6) 选中“Project”，新建一个项目(见图 7-30)。

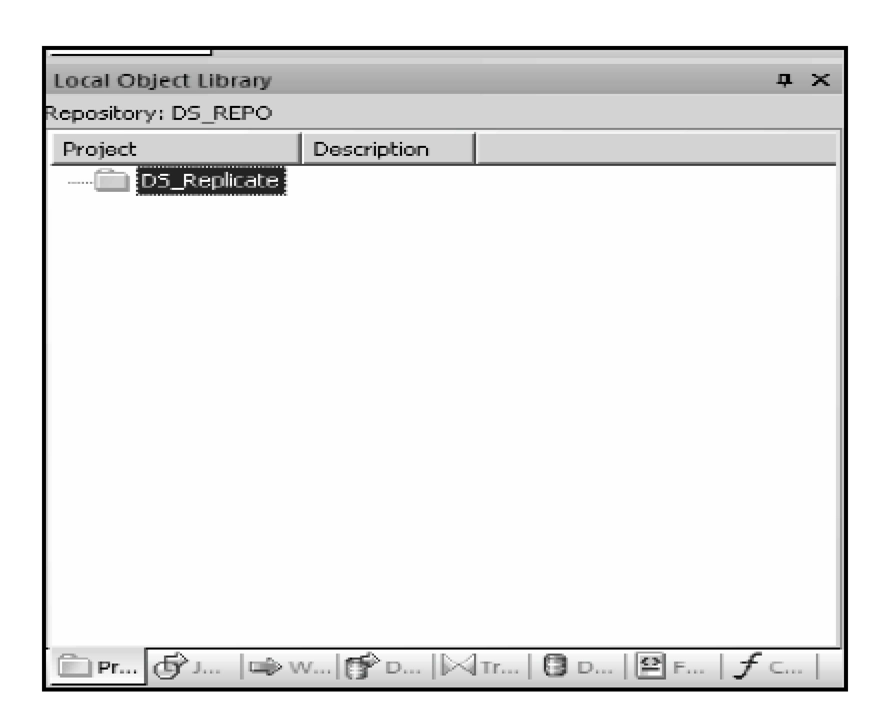


图 7-30

(7) 选中“Jobs”，新建一个工作(见图 7-31)。

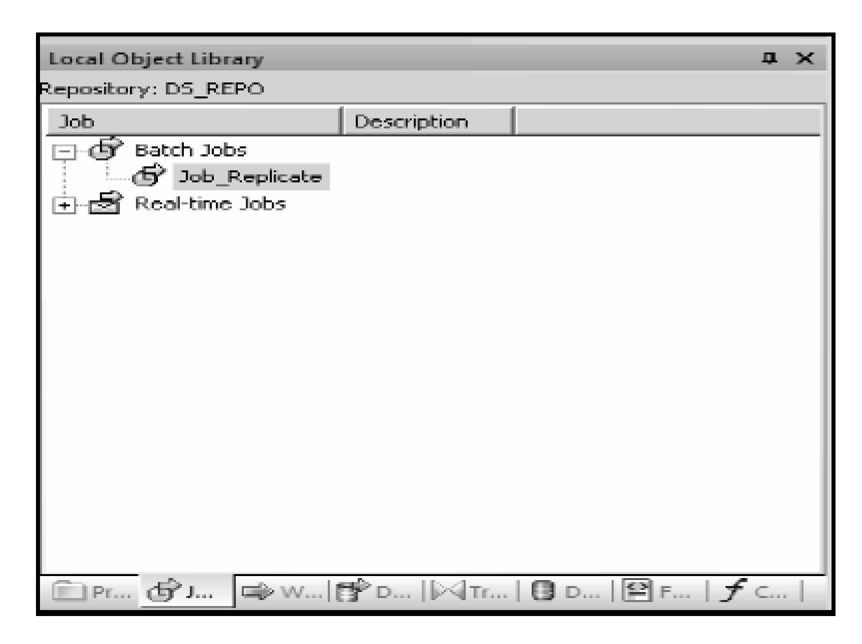


图 7-31

(8) 将新建立的工作拖至“Project Area”面板项目之下(见图 7-32)。

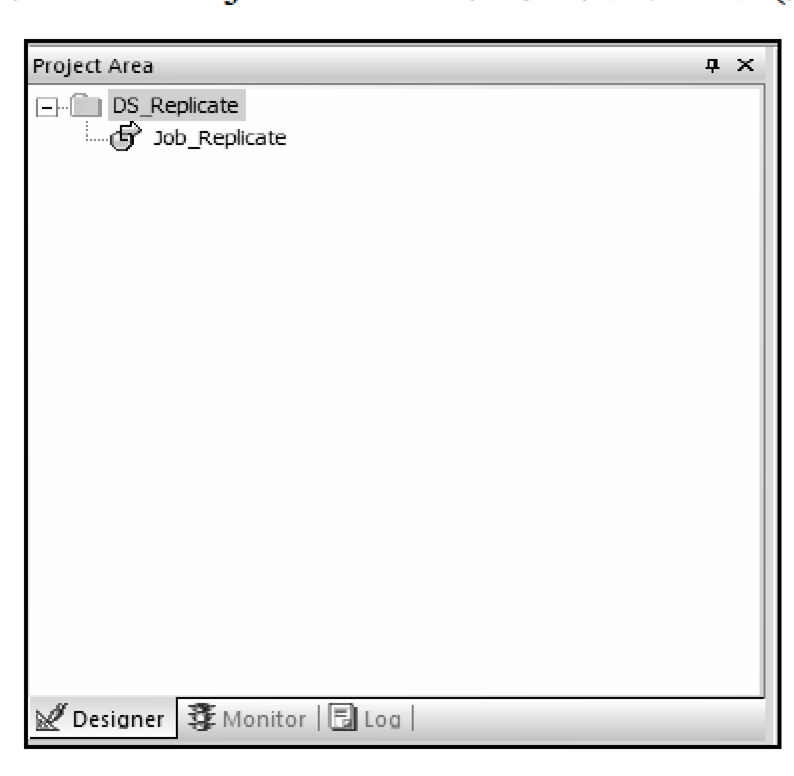
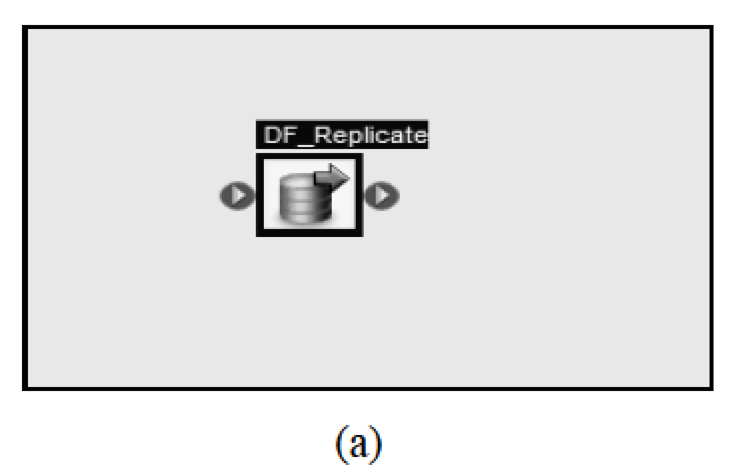
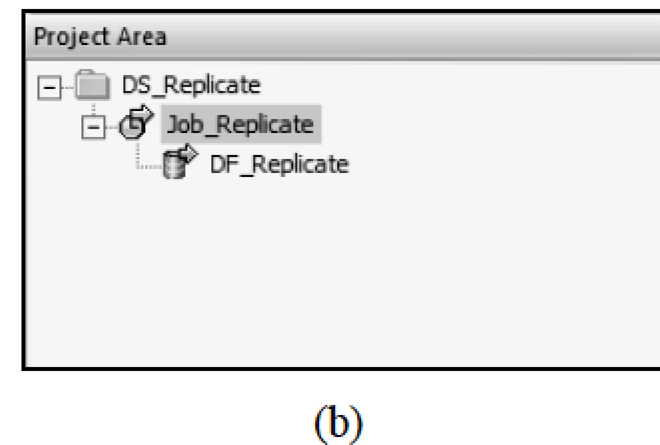


图 7-32

(9) 选中该工作，在“Data Service”最右边的工具栏中选中[图标]，并拖至中间工作面板中。如图 7-33 所示，可以看到该数据流已自动绑定至工作下面。



(a)



(b)

图 7-33

(10) 继续在出现的界面左下方，选中“Data Store”，新建“SAP HANA Data Store”(见图 7-34)。

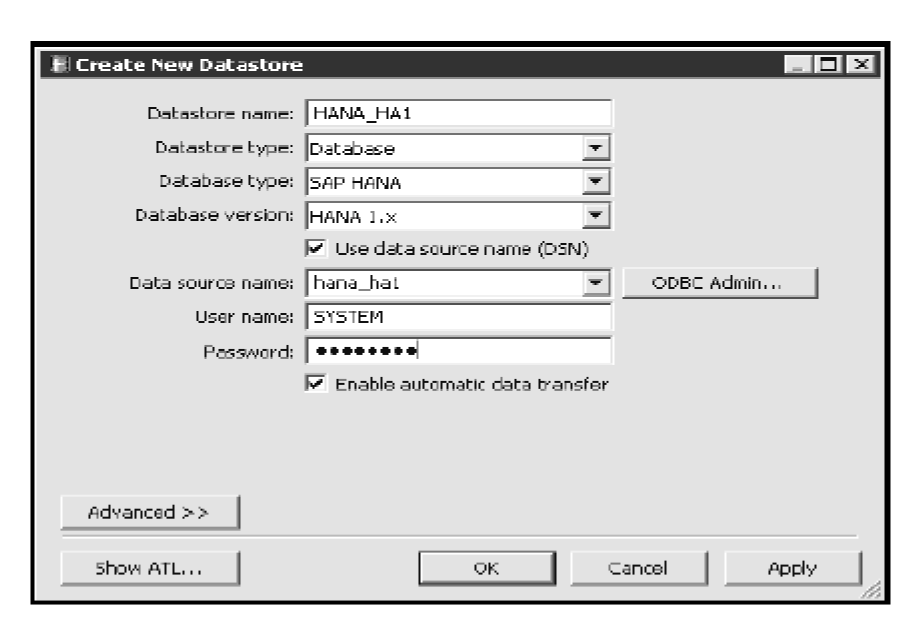


图 7-34



(11) 重复上一步新建一个源数据的“Data Store”。

(12) 展开新建的源数据“Data Store”，双击“Tables”项，可以看到其中所有表(见图 7-35)。

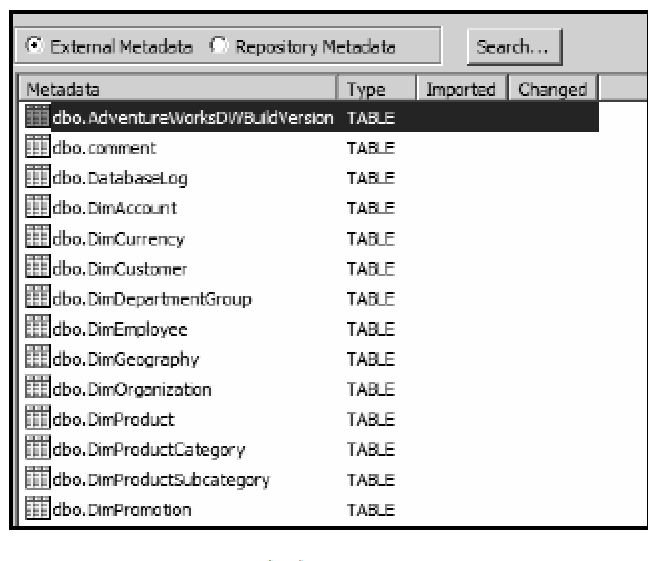


图 7-35

(13) 鼠标右击选中需要复制数据的表，选择“Import”选项(见图 7-36)。

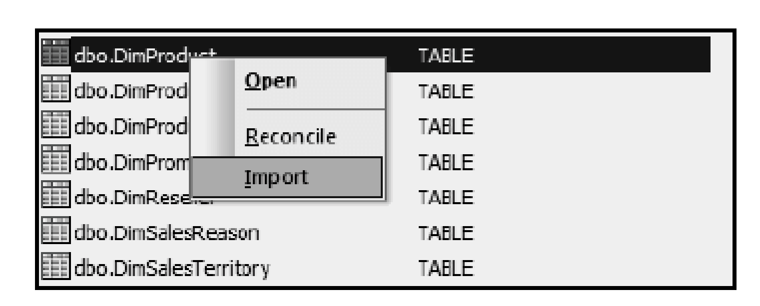


图 7-36

(14) 可以看到该表已被导入“Data Store”的选项表(见图 7-37)。



图 7-37

(15) 双击第 9 步中创建的工作流，将上步中导入的表拖曳至工作面板，选择“Make Source”。然后在最右边的工具栏中拖曳至工作面板。最后选中“SAP HANA Data Store”中的“Template Tables”，并将其拖曳至工作面板中。用鼠标将它们的接口相连，如图 7-38 所示。

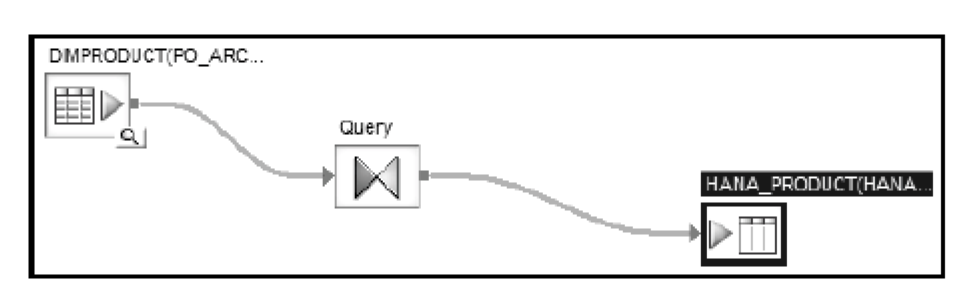


图 7-38

(16) 双击“Query”节点，将“Schema In”中的所有列拖曳至 Schema Out 中(见图 7-39)。

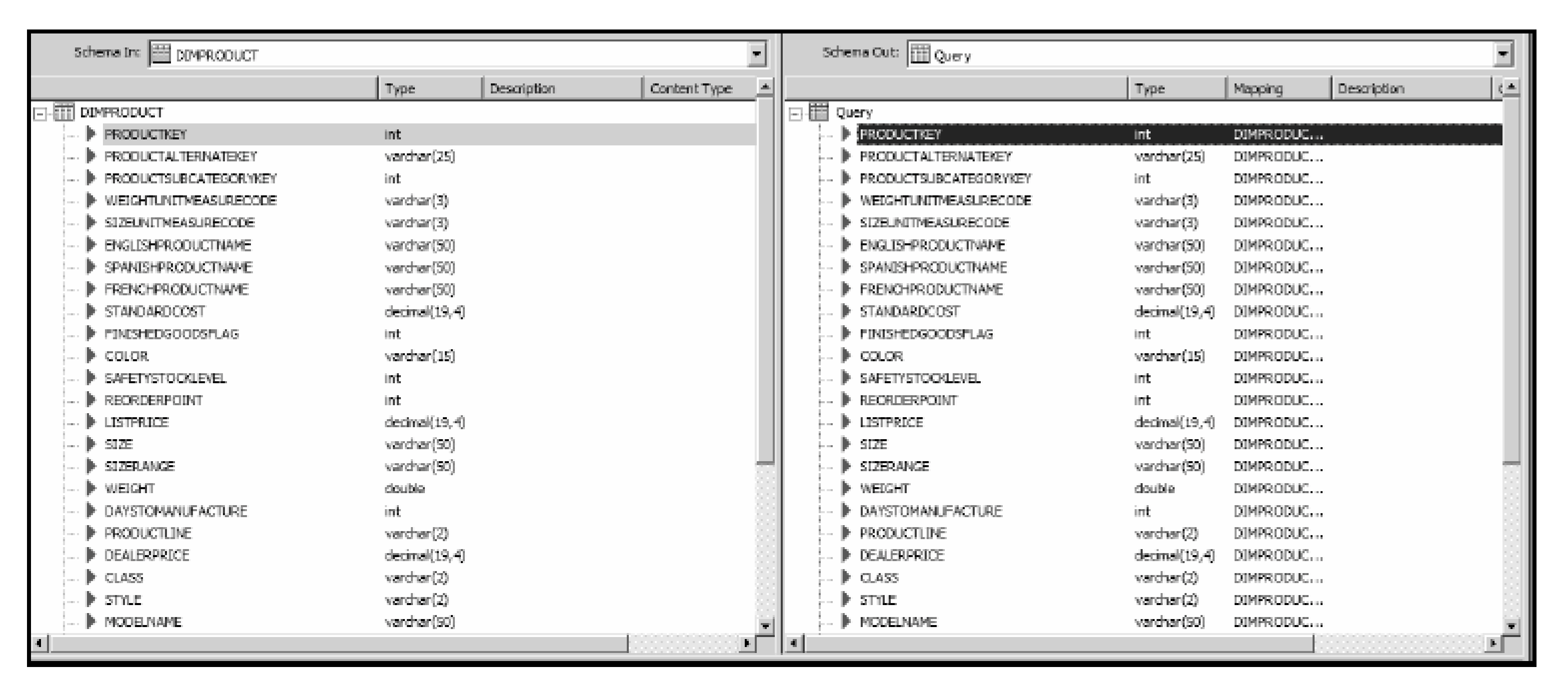


图 7-39



(17) 至此，我们所有的复制配置已完成。单击“Save All”并右击“Job”节点选择“Execute”，我们可以看到工作被顺利执行(见图 7-40)。

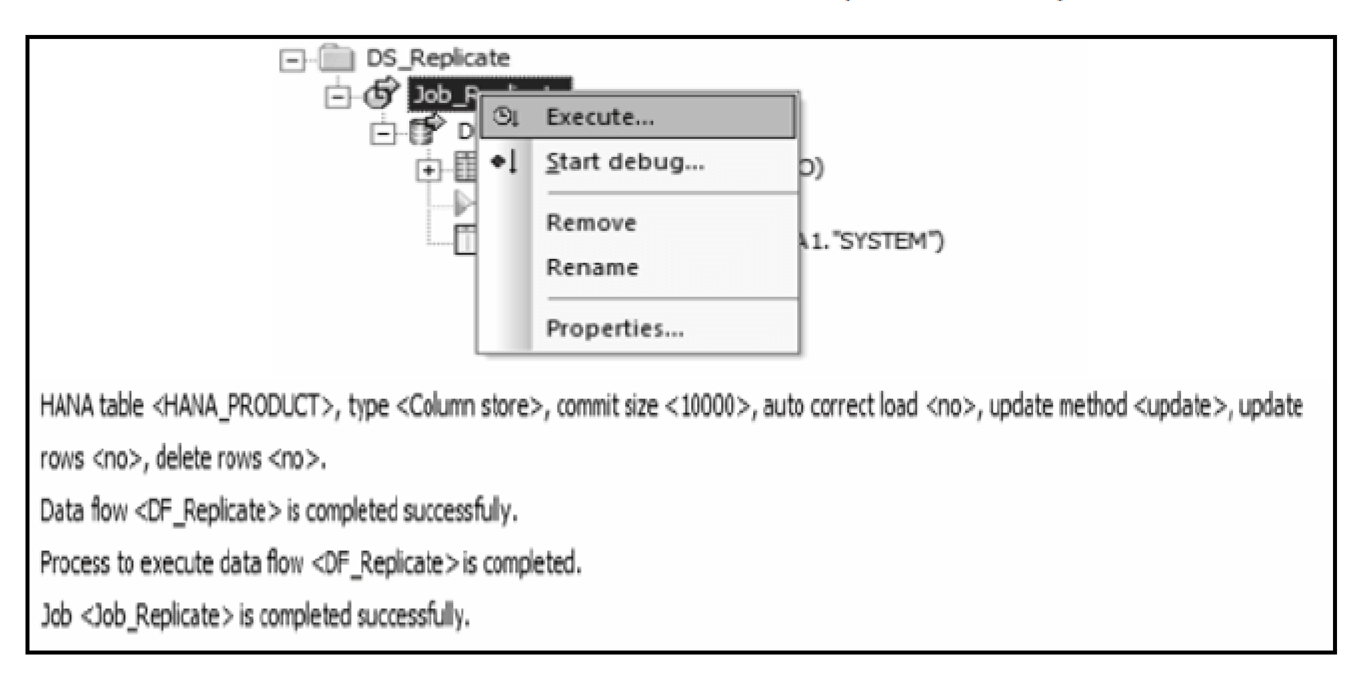


图 7-40

(18) 打开 SAP HANA Studio，检查相应的表是否已在 SAP HANA 中创建并且数据已经被成功复制。

本章小结与练习

在本章中，我们介绍了几种将数据导入 SAP HANA 的方法。包括：

- 本地文本文件导入
- CTL 方式
- 基于日志的复制(Sybase Replication)
- 基于 ETL 的复制(Data Service)
- 基于触发的复制(SLT Replication)
- 提取器直接抽取(Direct Extractor Connection)

随后我们对其中的本地文本文件导入、CTL 方式、SLT 与 ETL 方式给出实际操作来进一步讲述。需要记住的是 SLT 在实时性方面的优势以及 Data Service 的强大功能性。

练习

自己创建 CSV 文件，并将其中数据导入 SAP HANA 的数据表。

第八章　在 SAP HANA Studio 中建立数据模型

在本章里，我们将详细介绍如何在 SAP HANA Studio 中建立数据模型。SAP HANA 的建模，是指通过生成一系列的视图来对数据库表单内的数据进行数据精炼或数据切片以描述一种业务模式的行为。通过建模生成的视图可以用于报表和决策，我们在 SAP HANA 中统称此类视图为信息视图(information views)。信息视图通过使用对内容数据(content data)的各种组合来建立业务实例的模型。内容数据可以进行如下分类，即属性和度量(Attribute & Measure)。属性通常是指描述类的数据，例如客户名称、城市、邮编等数据；度量通常是指可计量的数据，例如年利润、季度销售额、采购数额等数据。

在 SAP HANA Studio 中，我们是通过 Modeler 透视图来对基础数据进行建模操作的，Modeler 透视图提供了非常有用的图形化工具来帮助你建立和编辑数据模型以及相关的存储过程。与此同时你能通过这些图形化工具生成用于分析的权限来限制用户对系统内数据模型的访问，即根据用户所分配的权限的不同，他们能看到的数据模型也不相同。在 Modeler 透视图中我们能够创建的信息视图包括属性视图(Attribute Views)、分析视图(Analytic Views)和计算视图(Calculation Views)。

第一节　基础概念

一、属性和度量

在正式开始建立数据模型之前，我们先简单介绍一些在 SAP HANA Studio 建模时需要用到的基础概念。前面我们讲过，用于建立模型的内容数据分为属性和度量两种类型。下面我们来详细讲述一下这两种数据类型。



1. 属性

属性(Attributes)是一种独立的用于分析的元素，这种数据元素是没有度量性质的描述类数据。

(1) 简单属性数据

简单属性数据就是直接从数据库表中获取的没有度量性质的数据，例如产品ID、产品名称。

(2) 计算属性数据

计算属性数据是由已经存在的一个或者多个简单属性数据或常量组成的数据。例如产品的“全称”就是计算属性数据，它是由“产品名称”+“产品版本号”两个简单属性数据组合而成的。以 iPhone 产品为例，“iPhone5S”(产品全称) = “iPhone”(产品名称)+“5S”(产品版本号)。

(3) 私有属性数据

私有属性数据是用来让你在分析视图中为该视图单独定制某一属性的。例如，如果分析视图或者计算视图中引用了某个属性视图，那么该属性视图的所有属性数据的行为都会被分析视图或者计算视图继承。即在属性视图中更改了任意属性数据，都会影响所有引用此属性视图的其他分析视图或者计算视图。如果你不想让分析视图中某些属性数据的行为根据属性视图的更改而更改，就要为此分析视图定义单独的私有属性数据。定义完私有属性数据后，无论分析视图引用的属性视图如何变化，该私有属性数据都不会改变。



2. 度量

度量(Measures)是一种提供度量功能的分析性数据元素。这类数据通常用于分析视图和计算视图。

(1) 简单度量数据

简单度量数据提供度量功能的数据元素，例如利润。

(2) 计算度量数据

计算度量数据是在来自 OLAP 信息立方体的数据、计算操作、常量和功能的组合上定义的。例如，计算度量数据可以用来计算一个产品跨 5 个地域的销售汇总情况。计算度量数据也可以给一个计算分配常量值。

(3) 限制度量数据

限制度量数据用来在用户自定义规则的基础上过滤一些数值。例如我们只让价

格高于 100 元的产品参与到计算视图或者分析视图的数据模型中，价格低于 100 元的产品数据则被过滤掉。

(4) 计数器

计数器(Counters)表示在计算视图中对一个属性数据重复出现的次数。例如，产品出现的次数。

二、属性视图、分析视图和计算视图

1. 属性视图

属性视图是基于不同的源数据库表中具有一定关系的属性数据而建立起的实体模型。例如，客户 ID 是一种属性数据，用于描述是谁买了产品。然而，对于一个客户来说，系统里有更多数据来描述这个客户的属性，如客户地址、客户状态、客户关系情况等，这些数据都可以通过和客户 ID 在数据库中对所属的数据表进行连接查询(Join)而得到。基于此，用户可以创建一个关于客户的属性视图。该视图将来自不同数据库表的与客户相关的属性数据关联起来，以满足业务上的需求。

在属性视图中你能基于以下元素建立模型：

- 列(Columns)
- 计算列(Calculated Columns)
- 层级结构(Hierarchies)

你还可以在属性视图中通过设置如下属性来微调属性视图的特征。

- 通过设置过滤器(Filters)来限制用户使用属性视图时所选中的数据。
- 属性可以设置成“隐藏”。这样，虽然该属性的数据依然会被系统处理，但对于最终用户来讲，在结果中数据是不可见的。
- 属性类数据可以被定义成主键属性数据(Key Attributes)，并可用于与多个数据库表连接。
- 启用“Drill Down Enabled”功能后，属性数据可以下钻得到更多的相关信息。

建立完属性视图后，该属性视图可以通过与在分析视图或者计算视图中定义的



包含度量数据的数据库表连接来生成基于 SAP HANA 数据的星形模式①(Star Schema)。

2. 分析视图

分析视图是用来建立包含度量数据的模型。例如，代表销售订单历史数据的业务数据集市包含了数量、价格等度量数据信息。分析视图的数据基础是建立在多张数据库表之上的。在分析视图中被选用的度量数据必须来自这些数据库表中的一个。在一种情况下，分析视图可以是由属性数据和度量数据简单组合起来的，同时也可以是由属性视图和度量数据组合而成，在这种情况下，你可以对被引用的属性数据进行深度挖掘。对于在分析视图定义阶段所引用的属性视图，分析视图会继承它们的所有属性定义。

在分析视图中你能基于以下元素建立模型：

- 列
- 计算列
- 限制列(Restricted Columns)
- 变量(Variables)
- 输入参数(Input parameters)

你还可以在分析视图中通过设置如下属性来微调分析视图的特征。

- 通过设置过滤器来限制用户使用分析视图时所选中的数据。
- 属性数据可以设置成“隐藏”。这样，虽然该属性数据依然会被系统处理，但对于最终用户是不可见的。
- 属性数据可以被定义成主键属性数据，并可用于多个数据库表连接。
- 启用“Drill Down Enabled”功能后，属性数据可以下钻得到更多的相关信息。
- 对于度量数据可以使用聚合功能(aggregation)。
- 可以设置分析视图中货币和度量单位参数。



---

① 星形模式是一种多维的数据关系，它由一个事实表(Fact Table)和一组维表(Dimension Table)组成。每个维表都有一个维作为主键，所有这些维的主键组合成事实表的主键。事实表的非主键属性称为事实，它们一般都是数值或其他可以进行计算的数据；而维大都是文字、时间等类型的数据，按这种方式组织好数据我们就可以按照不同的维(事实表主键的部分或全部)来对这些事实数据进行求和(summary)、求平均值(average)、计数(count)、百分比(percent)的聚集计算，甚至可以做 20～80 分析。这样就可以从不同的角度来分析业务主题的情况。

3. 计算视图

计算视图可以对 SAP HANA 数据库中的数据定义更多高级切片功能。计算视图可以简单地用做和属性视图或分析视图同样的功能，但更多的是被用来满足业务应用上的一些复杂逻辑。这些逻辑是属性视图和分析视图无法实现的。

例如，计算视图有计算逻辑的层次概念，可以包含来自多个数据库表的度量数据，可以包含复杂的 SQL 逻辑等。计算视图的数据基础可以是包含多种数据库表或者字段的属性视图和分析视图的组合。在计算视图中用户可以实现数据库表的连接、联合、投影以及在源数据上的聚合功能。

在计算视图中你能基于以下元素建立模型：

- 属性数据
- 度量数据
- 计算度量数据
- 计数器
- 层级结构
- 变量
- 输入参数

计算视图可以包含度量数据，从而满足多维度报表的需求，也可以不包含度量数据而只用做列表形式的报表。创建计算视图时既可以使用图形编辑界面，也可以使用 SQL 编辑界面。这些多种多样的选项为复杂的业务需求提供了最大程度的灵活性。

三、建模决策树

介绍完以上三种信息视图后，我们在未来进行数据建模操作中就要考虑针对不同的情况，哪种视图是我们最合适的选择，因为根据我们选择视图的不同，系统的性能也会受到不同程度的影响。为了使系统性能达到最优化，我们必须严格遵循 SAP HANA 的建模决策树。在讲述建模决策树前，我们先来看看 SAP HANA 引擎的概览(见图 8-1)。

由图 8-1 我们可以看出，SAP HANA 为信息模型提供了不同类型的引擎：

- 连接引擎(Join Engine)用来处理所有数据连接的操作。
- OLAP 引擎(OLAP Engine)用来处理基于星形模式的数据计算和聚合操作。



图 8-1

- 计算引擎(Calculation Engine)用来处理复杂的计算操作，这些操作是连接引擎和 OLAP 引擎所不能够处理的。
- SQL 优化控制器(SQL Optimizer)的作用是根据请求的模型或者查询来调用合适的引擎来处理数据。

当我们创建信息模型时，我们需要考虑哪个引擎将被调用，图 8-2 显示了 SAP HANA 各个引擎和信息视图之间的关系，即我们在设计信息视图时，不同的信息视图会调用不同的引擎。有一点是需要着重指出的，任何包含计算属性数据的分析视图或者属性视图都会被视计算视图而被计算引擎处理。因此我们在设计有计算属性的信息视图时要特别小心，因为不同引擎的调用对系统的性能会产生很大的影响。通常来讲，我们需要尽量避免分析视图或属性视图带有计算属性的数据，这样我们就能让正确的引擎来处理正确的视图。

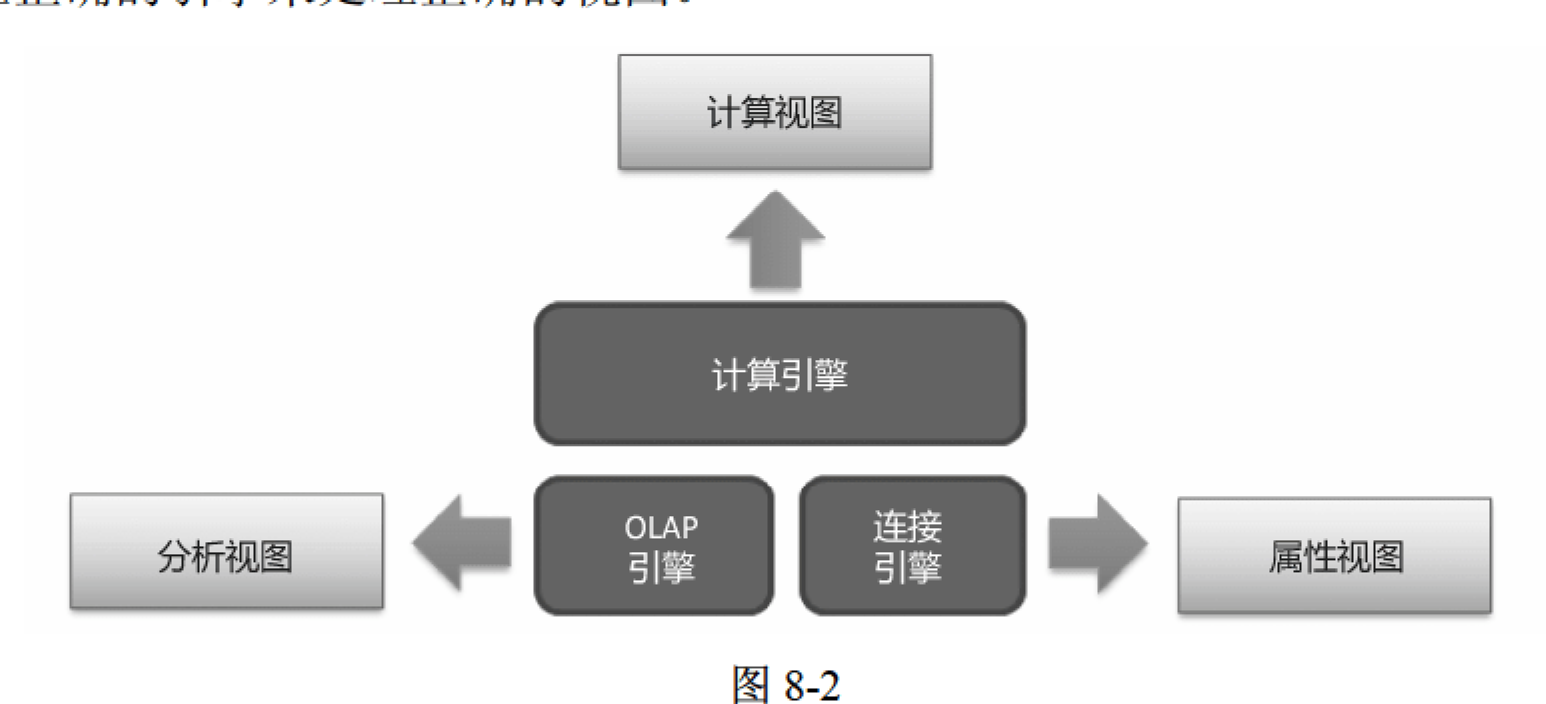


图 8-2

知道了不同的信息视图与 SAP HANA 引擎之间的关系之后，我们来看看表 8-1。表 8-1 列出了不同信息视图之间的用途和优缺点，我们在建模前可以使用这个表格来为我们选择信息视图进行参考。

表 8-1

	列数据表	分析视图	计算视图(SQL)	计算视图(CE)
用途	简单容易上手，特别适合于简单的程序和一般展示用途	特别适合用于分析目的的报表或应用，尤其是对大量数据进行只读操作时尤为适用	能够快速为需要复杂计算的业务需求建立模型，通常这类模型只包含很少的字段	特别适用于需要进行复杂运算的、分析视图无法完成的分析操作。
优点	不需要创建附加的模型，对于绝大多数终端程序都能非常容易地调用	支持建模操作并可以通过建模最优化，对于 SELECT 语句有很高的执行性能	使用 SQL 语言可以快速创建此类视图	终端查询操作能被最优化及支持并行处理，比基于 SQL 的计算视图性能好很多
缺点	不支持分析特权、多语言和终端控制。需要终端程序来进行复杂的逻辑计算，总体来讲性能比较低	不能进行复杂的计算操作	终端查询无法得到最优化处理，因此执行性能要远差于其他视图	其语法与常见的 SQL 语法有很大差别，需要花时间适应

与此同时，SAP HANA 还提供了决策树(Decision Tree)来指导我们如何选择最有效的用于建立模型的信息视图(见图 8-3)。我们建议大家在为某种业务决策而进行建模操作时，严格遵循图 8-3 所示的决策树来确定最优化及最正确的信息视图。有一点需要指出的是，为了使信息模型达到最好的性能并能最方便地维护已建立的信息模型，我们需要尽可能地只使用前三层决策树所引用的信息视图，即分析视图、属性视图和图形化计算视图。

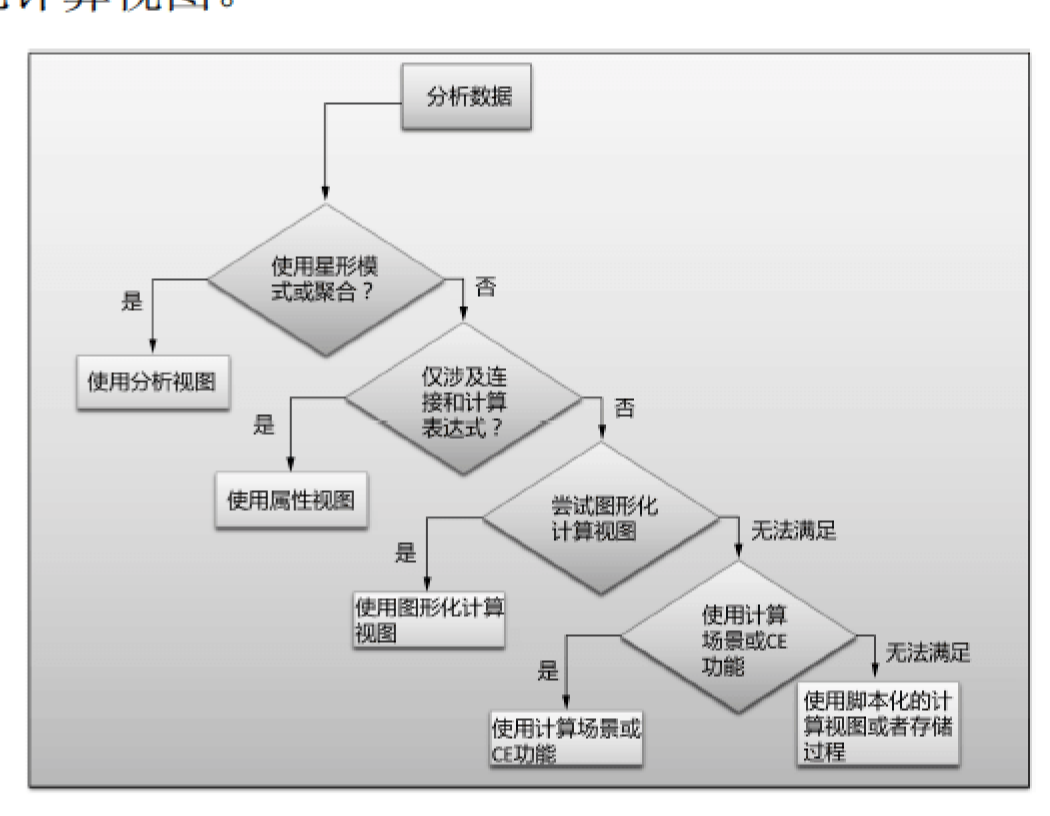


图 8-3



第二节　SAP HANA Studio 之数据表操作

对数据表进行操作是 SAP HANA 建模操作中最基本，也是最重要的操作之一。因此我们首先要熟悉如何在 SAP HANA Studio 中对于数据表进行创建、连接等基本操作。

一、建立数据表

首先我们先来尝试通过 SAP HANA Studio 创建第一个数据表。在本例中我们以建立名为“CUSTOMER”(客户表)的数据表为例，具体创建的方式如下。

(1) 打开 SAP HANA Studio，在“SAP HANA Systems”视图中展开所要创建数据表的系统节点，在本例中我们使用系统“HA1”，然后定位到“Catalog”→“SAP_HANA_EPM_DEMO”→“Tables”，如图 8-4 所示。

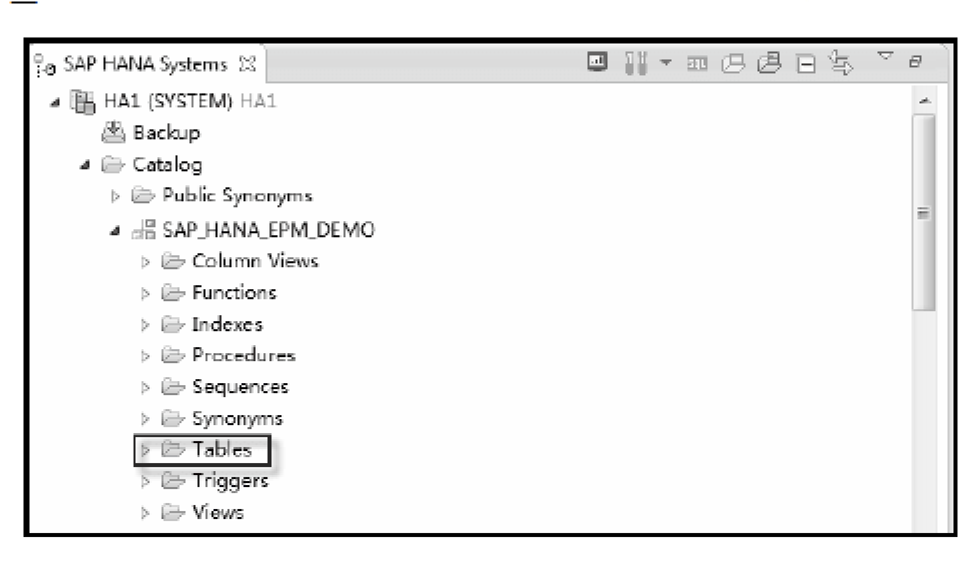


图 8-4

(2) 右击“Tables”文件夹，选择“SQL Console”项，打开 SQL 控制台视图(见图 8-5)。

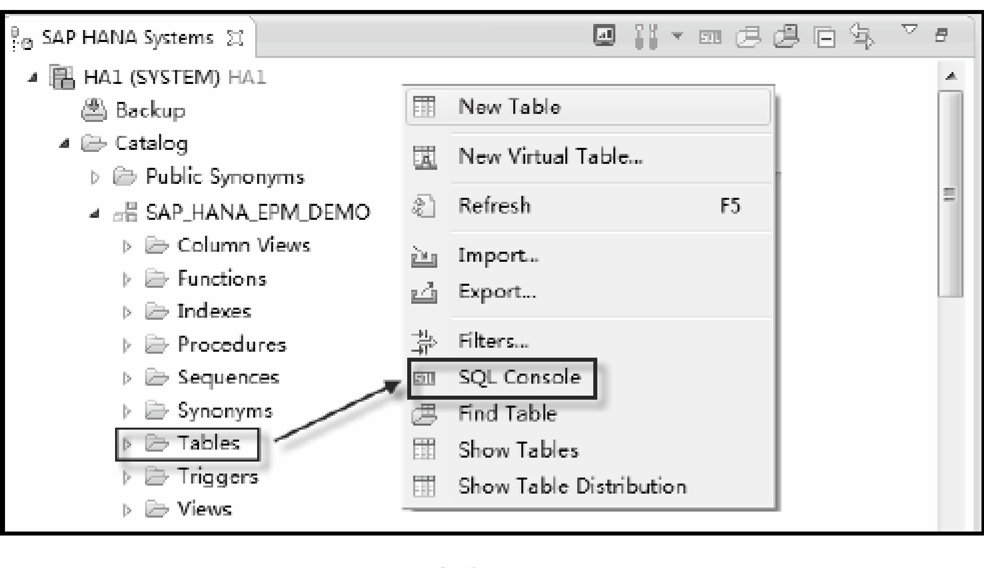


图 8-5



(3) 在 SQL 控制台视图中的空白处输入如下语句并单击“执行”(F8)，如图 8-6 所示。

```
create column table "SAP_HANA_EPM_DEMO"."CUSTOMER"
( "CUSTOMER_ID" INTEGER not null,
       "CUSTOMER_NAME" NVARCHAR (40) null,
       "AREA_ID" INTEGER null,
       "PRODUCT_ID" INTEGER null,
       primary key ("CUSTOMER_ID"))
```

我们会在下一章详细介绍 SAP HANA SQL 语法，因此在这里我们暂时不对这些 SQL 语句做逐一的解释。

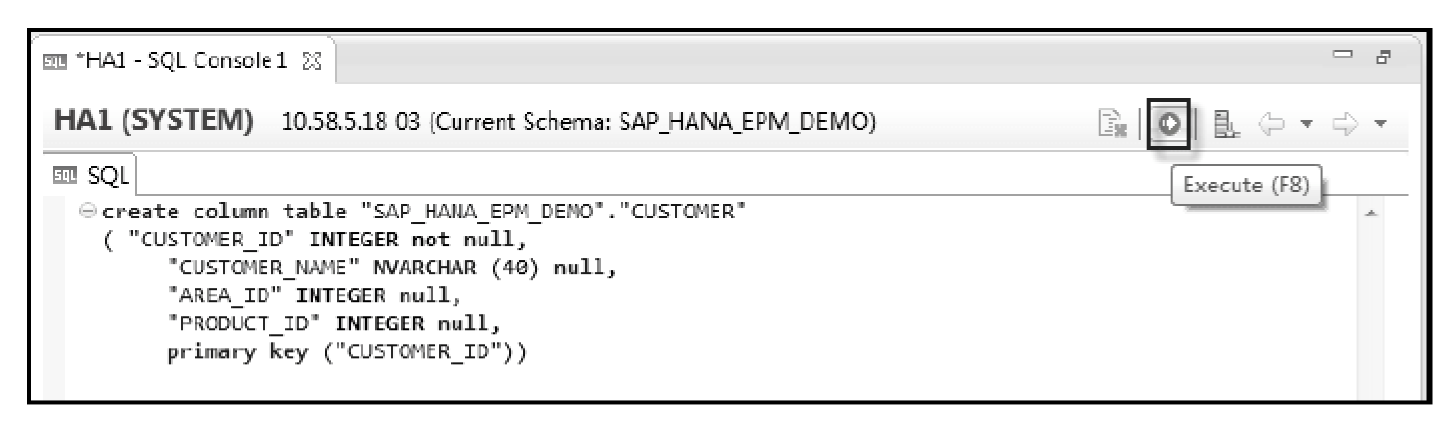


图 8-6

(4) 检查 SQL 语句执行结果(见图 8-7)。

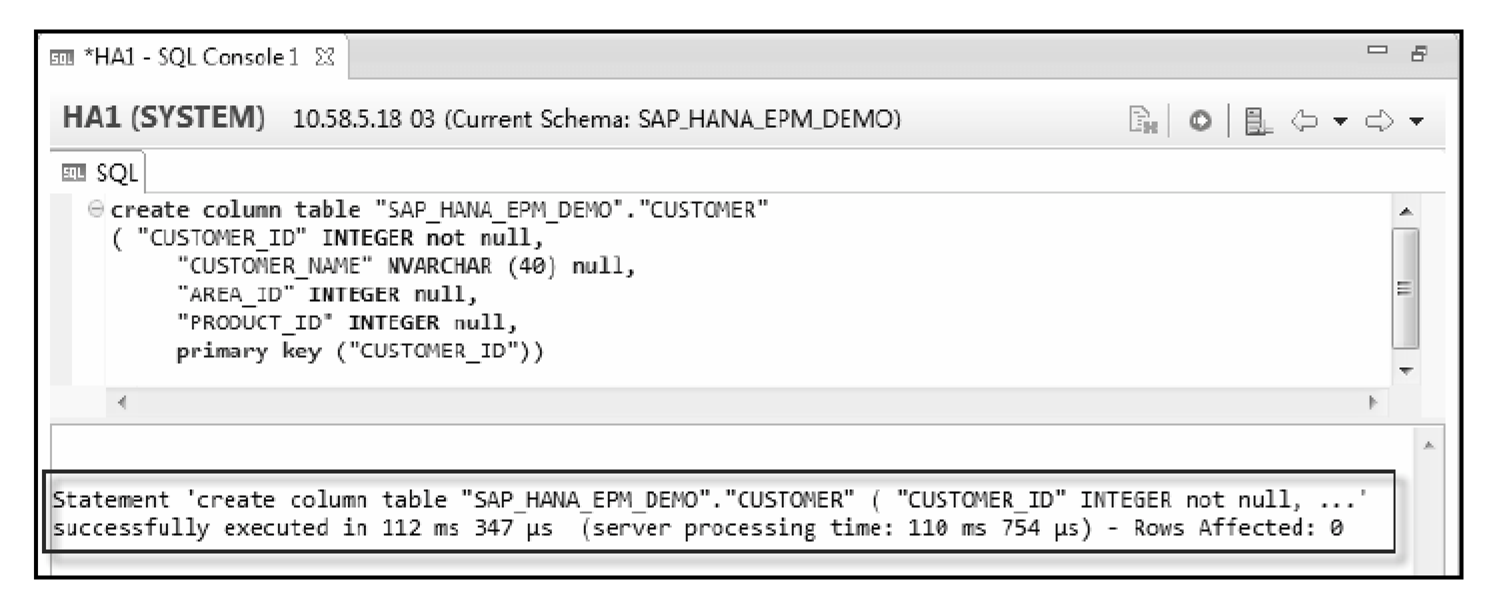


图 8-7

(5) 同理，我们可以依次创建“区域表”和“产品表”。

“区域表”：“AREA”

```
create column table "SAP_HANA_EPM_DEMO"."AREA"( "AREA_ID" INTEGER not n
ull,"AREA_DESCRIPTION" NVARCHAR (40) null,primary key ("AREA_ID"))
```

"产品表"："PRODUCT"

```
create column table "SAP_HANA_EPM_DEMO"."PRODUCT"( "PRODUCT_ID"
INTEGER not null,"PRODUCT_NAME" NVARCHAR (40) null,primary key
("PRODUCT_ID"))
```

(6) 最终生成的数据表结果如图 8-8 所示。

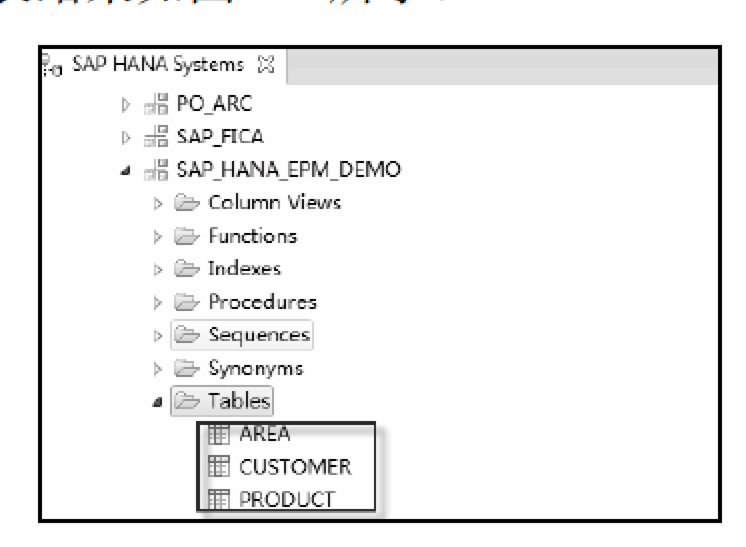
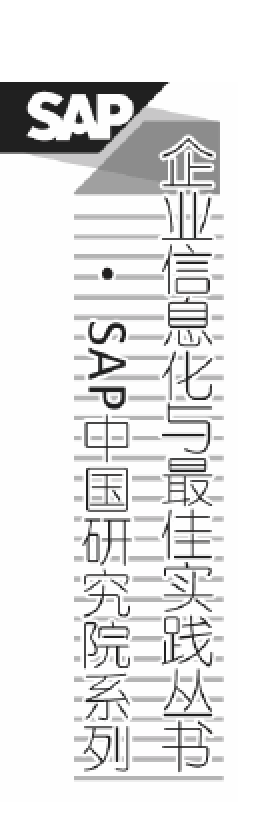


图 8-8

(7) 接下来，我们使用 INSERT 语句为各表添加数据。我们在第七章已经介绍了如何为 SAP HANA 数据库中的表导入数据。因为本例中涉及的数据条数很少，我们可以直接使用 INSERT 语句添加数据。请参照第 2 步打开 SQL 控制台视图，输入如下语句并"执行"(F8)来为各表添加数据，如图 8-9 所示。

首先我们插入"客户表"需要的内容：

```
insert into "SAP_HANA_EPM_DEMO"."CUSTOMER" values('100','客户甲','21','101');
insert into "SAP_HANA_EPM_DEMO"."CUSTOMER" values('200','客户乙','10','102');
insert into "SAP_HANA_EPM_DEMO"."CUSTOMER" values('300','客户丙','22','101');
insert into "SAP_HANA_EPM_DEMO"."CUSTOMER" values('400','客户丁','24','109');
```



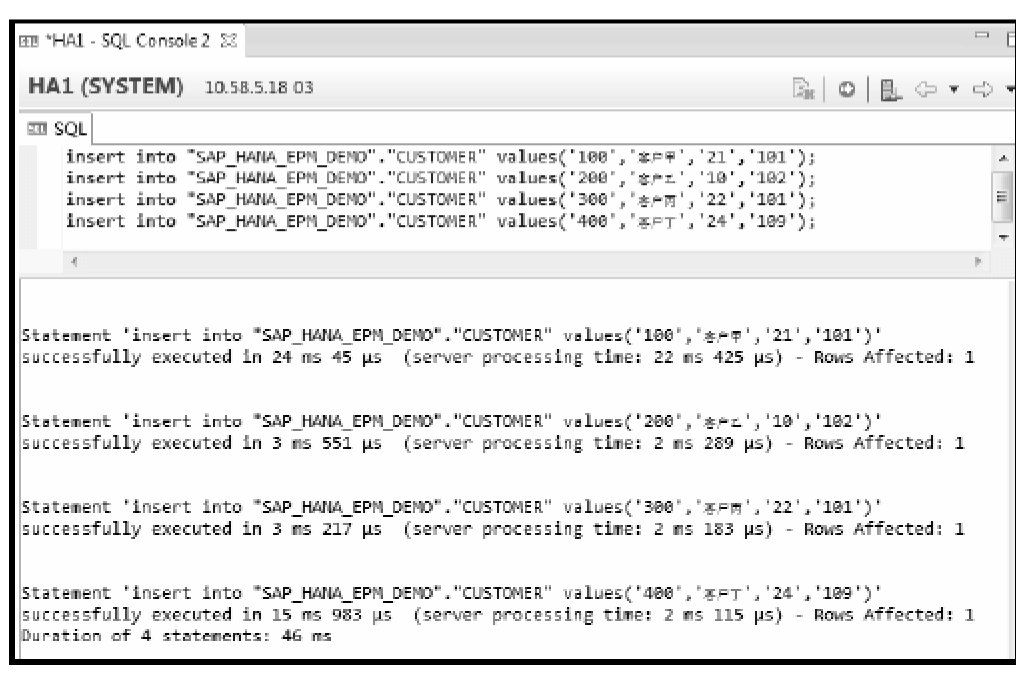


图 8-9

“区域表”

```
insert into "SAP_HANA_EPM_DEMO"."AREA" values('10','北京');
insert into "SAP_HANA_EPM_DEMO"."AREA" values('22','天津');
insert into "SAP_HANA_EPM_DEMO"."AREA" values('21','上海');
insert into "SAP_HANA_EPM_DEMO"."AREA" values('23','重庆');
insert into "SAP_HANA_EPM_DEMO"."AREA" values('25','南京');
insert into "SAP_HANA_EPM_DEMO"."AREA" values('27','武汉');
```

“产品表”

```
insert into "SAP_HANA_EPM_DEMO"."PRODUCT" values('101','键盘');
insert into "SAP_HANA_EPM_DEMO"."PRODUCT" values('102','鼠标');
insert into "SAP_HANA_EPM_DEMO"."PRODUCT" values('103','主板');
insert into "SAP_HANA_EPM_DEMO"."PRODUCT" values('104','显示器');
insert into "SAP_HANA_EPM_DEMO"."PRODUCT" values('105','内存');
```

(8) 执行完成后，在“客户表”数据表名称处右击→“Open Content”查看输入数据(见图 8-10)。

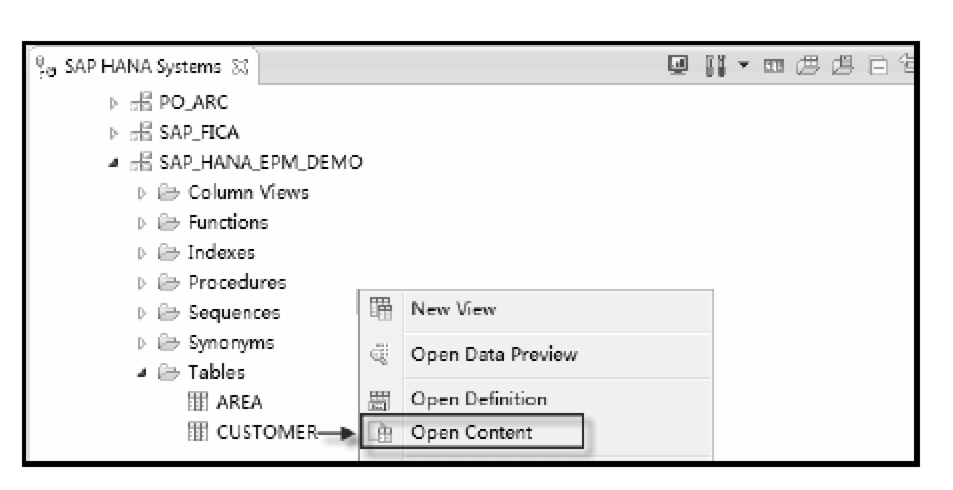
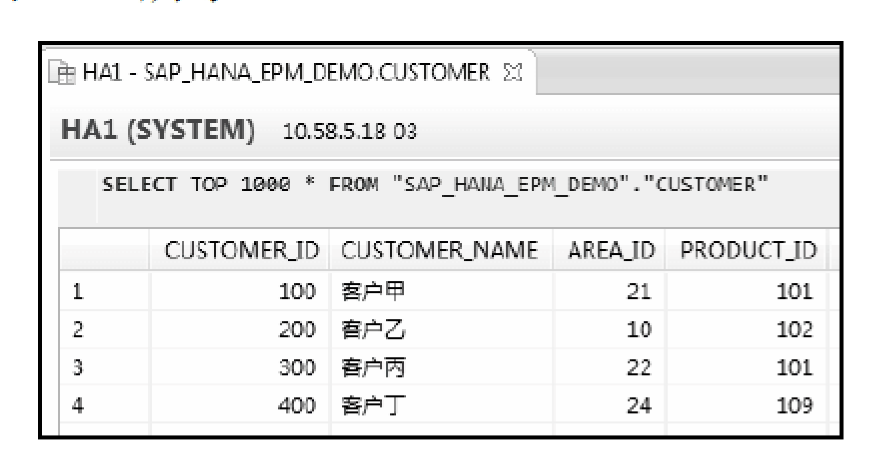


图 8-10

得到输出结果如图 8-11 所示。



	CUSTOMER_ID	CUSTOMER_NAME	AREA_ID	PRODUCT_ID
1	100	客户甲	21	101
2	200	客户乙	10	102
3	300	客户丙	22	101
4	400	客户丁	24	109

图 8-11

(9) 同理依次打开“区域表”和“产品表”来查看新生成的数据，如图 8-12 所示。

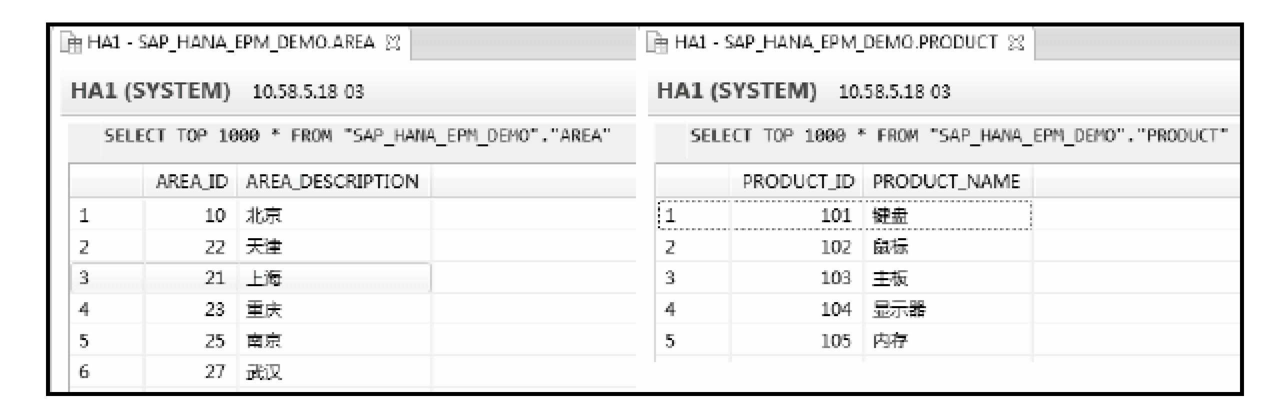


图 8-12

至此，我们新生成了三个数据表并为这三个数据表添加了测试数据，接下来我们将基于这些测试数据表和测试数据来逐个介绍不同的表连接方式以及连接所得到的结果比较。

二、连接数据表

对于数据表连接(Table Join)操作，相信对于懂 SQL 语言的读者应该是轻车熟路了。而我们接下来介绍的，是如何在 SAP HANA Studio 中完成数据表的连接操作。和原始的 SQL 语句不同的是，SAP HANA Studio 提供了图形化的工具来帮助开发者快速实现复杂的数据表连接操作。

下面我们以创建一个常见的内连接为例，看看在 SAP HANA Studio 中怎么样通过图形化的界面快速生成数据表连接视图和得到其连接结果。

(1) 打开 SAP HANA Studio，定位到“Catalog”→“SAP_HANA_EPM_DEMO”→“Tables”，单击展开“Tables”文件夹，在列出的任意表名称处右键单击，在调出的右键菜单中选中“Generate”→“Visual SQL”(见图 8-13)。

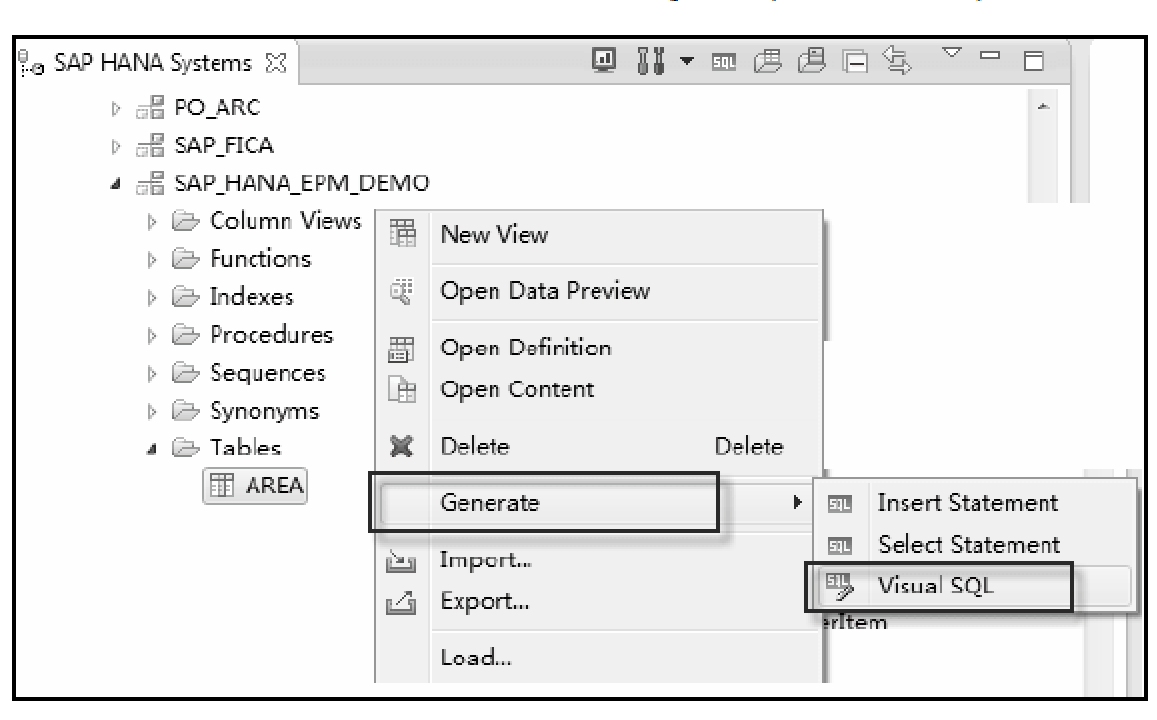


图 8-13

(2) 在可视化 SQL 控制台中，分别拖选“客户表”和“产品表”进到视图内。我们选择“PRODUCT_ID”作为连接条件，在“SAP_HANA_EPM_DEMO.PRODUCT”数据表的“PRODUCT_ID”字段按住鼠标左键不放，将鼠标箭头拖动到右侧“SAP_HANA_EPM_DEMO.CUSTOMER”数据表的“PRODUCT_ID”字段上，我们就完成了一个最简单的内连接操作，如第 3 步所示，只需将两表的“PRODUCT_ID”字段通过拖曳连接到一起即可(见图 8-14)。

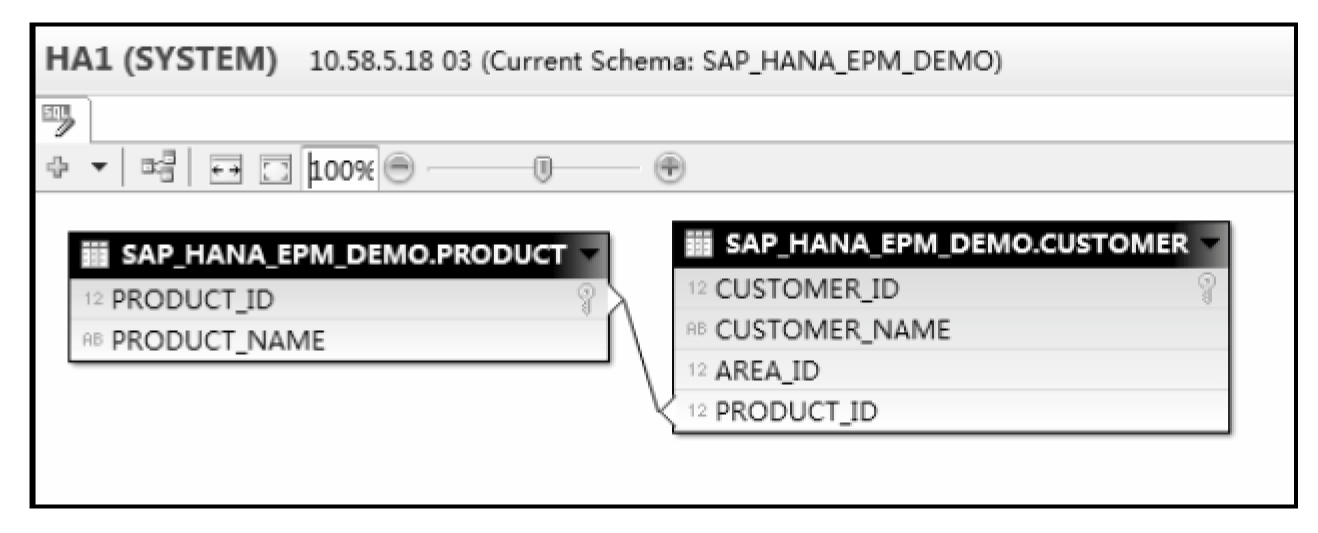


图 8-14

你可以通过双击“PRODUCT_ID”字段之间的连线，如图 8-15 所示，来调出“Insert join dialog”对话框以修改条件表达式。

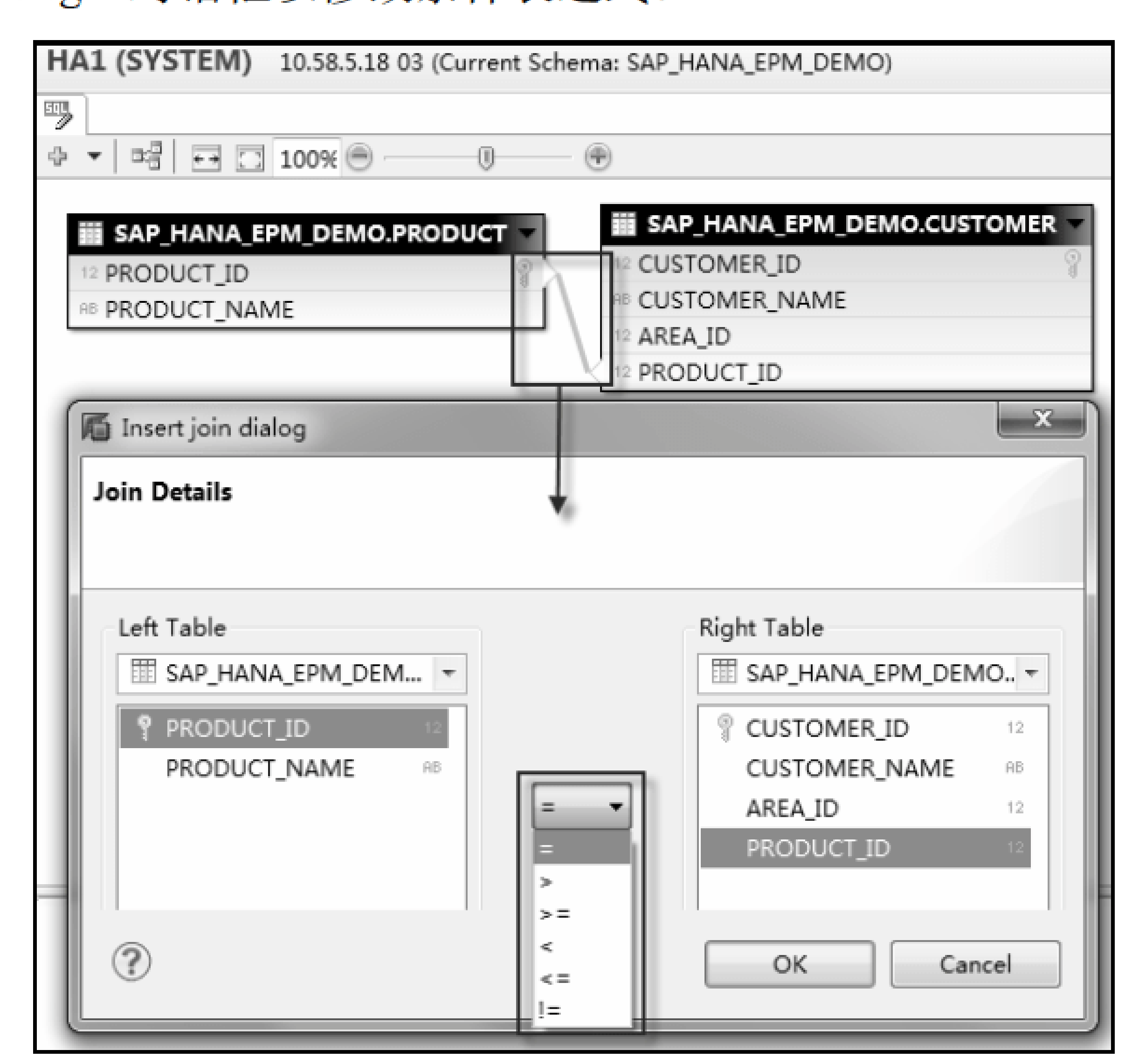


图 8-15

(3) 设好条件后，请依次将需要输出的字段拖曳到下方的空白区域。通常来讲，我们在输出视图中不需要把所有的字段都显示出来，因此我们可以在此自定义哪些字段是需要显示的。在这里我们选择三个字段“PRODUCT_ID”、“PRODUCT_NAME”和“CUSTOMER_NAME”显示在输出视图中，如图 8-16 所示。

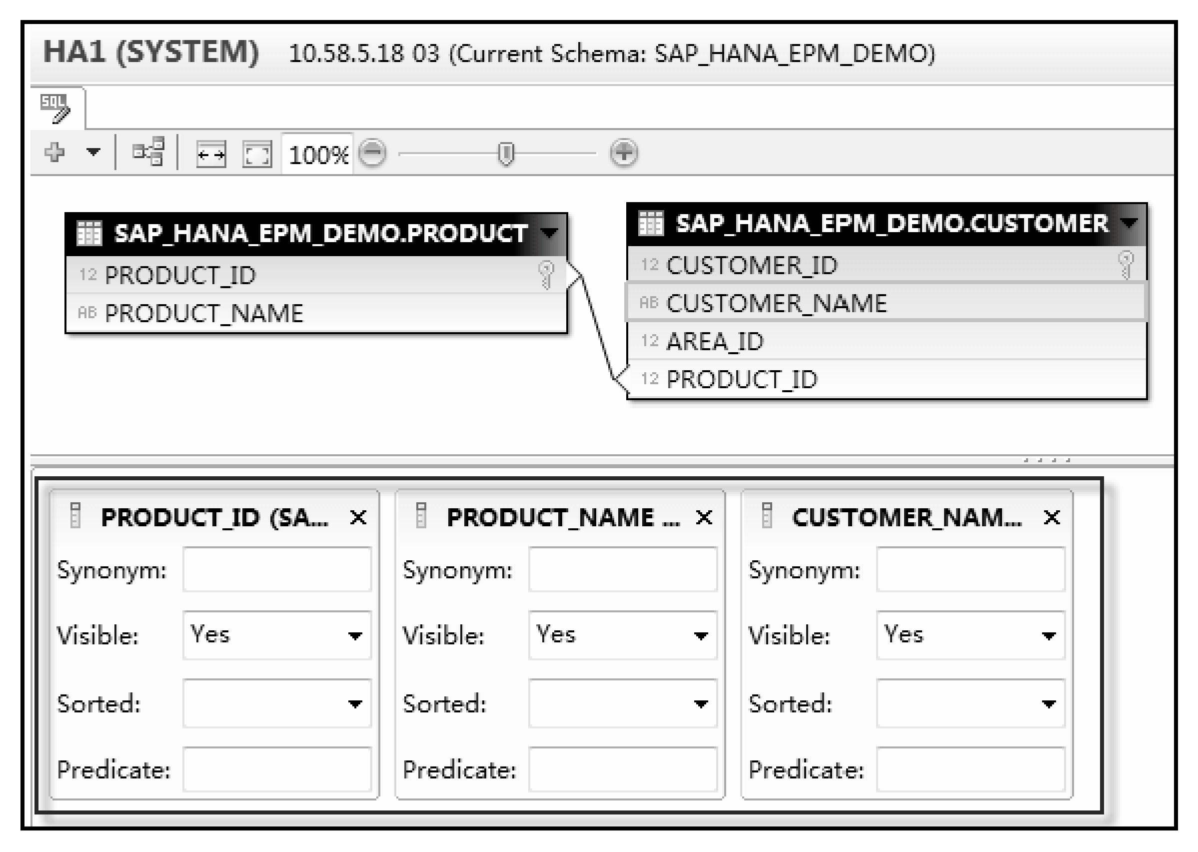


图 8-16

(4) 选定输出字段后，我们能对输出字段的可见性，排序等做更多调整，然后在可视化 SQL 控制台空白处右键单击，在弹出的右键菜单中选择“Execute”，如图 8-17 所示。



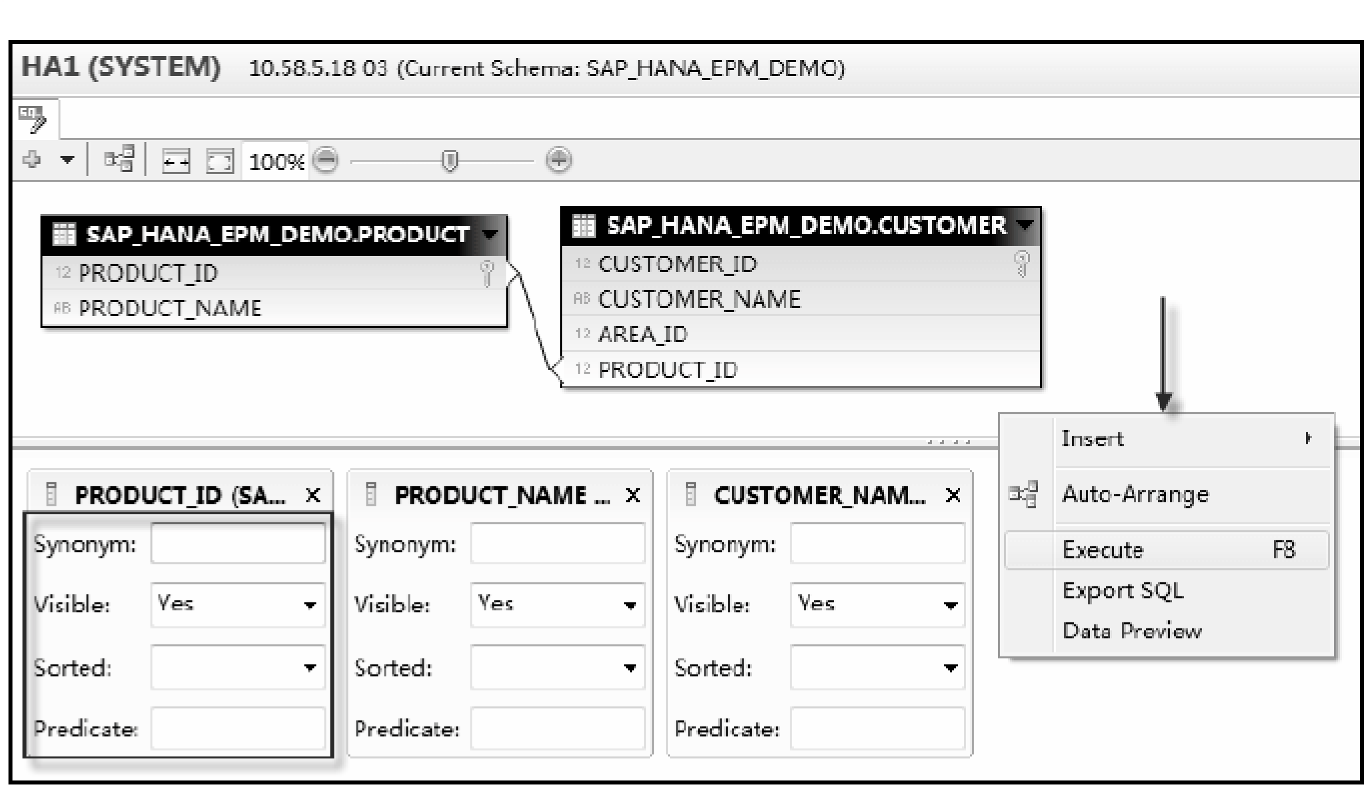
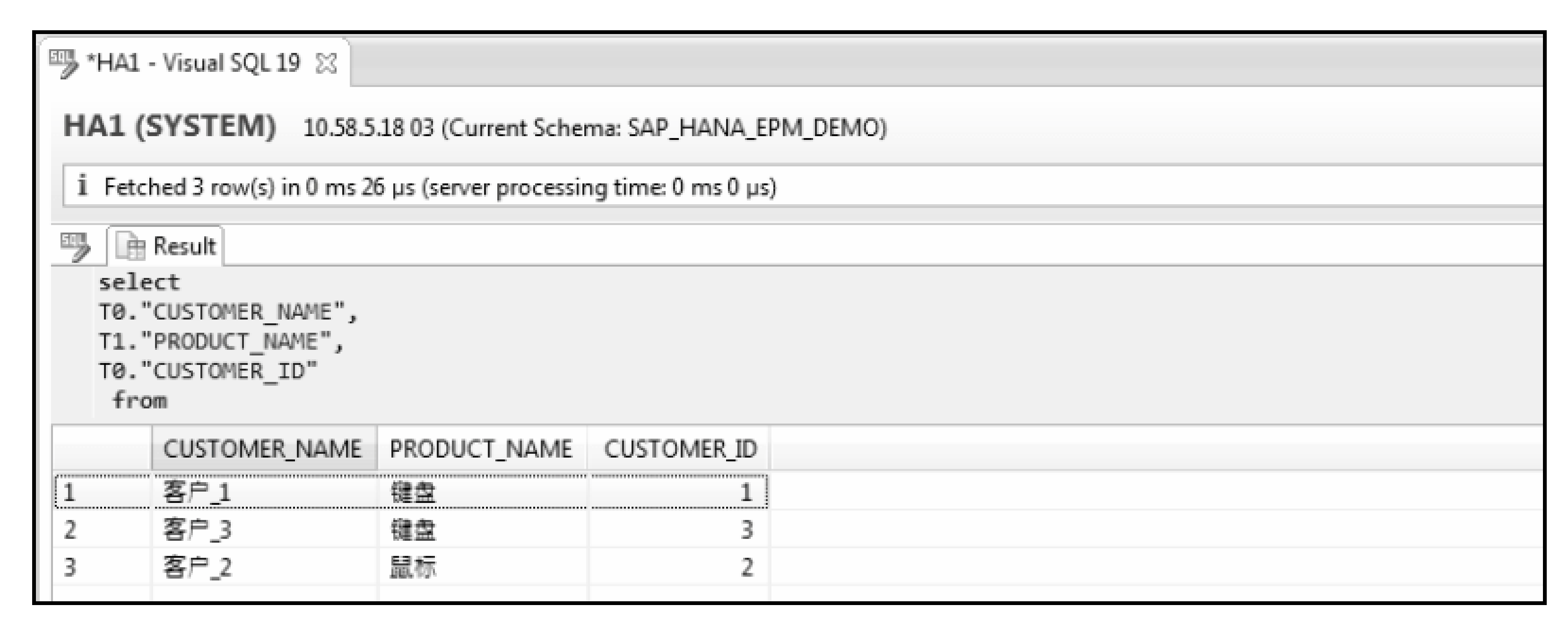


图 8-17

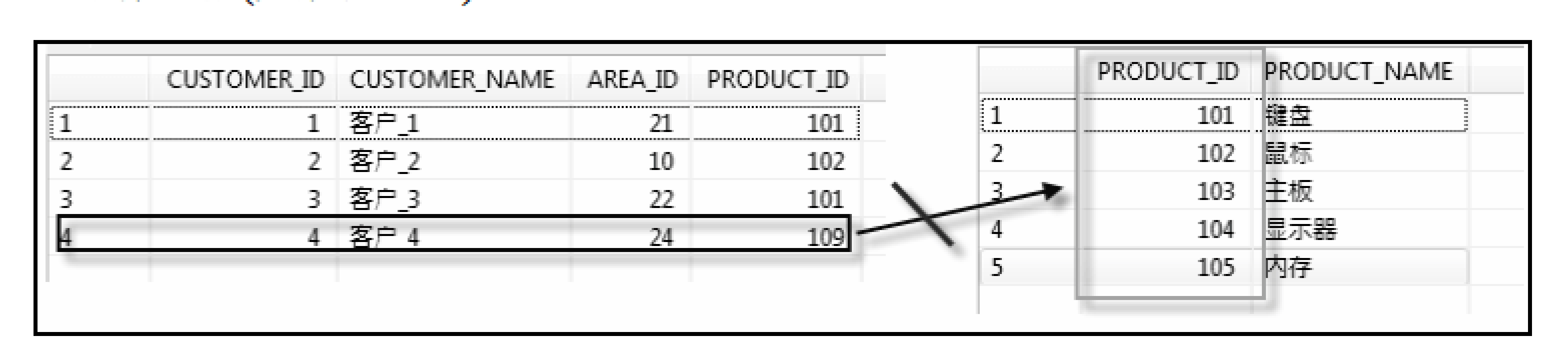
执行成功后，我们能得到如图 8-18 所示的输出视图。



	CUSTOMER_NAME	PRODUCT_NAME	CUSTOMER_ID
1	客户_1	键盘	1
2	客户_3	键盘	3
3	客户_2	鼠标	2

图 8-18

我们知道，内连接的特点是从结果表中删除与其他被连接表中没有匹配行的所有行，即返回的结果集是两个表中所有相匹配的数据，而舍弃不匹配的数据，所以内连接的结果可能会丢失信息。在上图所示的结果表中，你会发现“客户_4”这条记录被删除了，这是因为在“产品表”中没有“客户_4”购买的“PRODUCT_ID”为“109”的产品(见图 8-19)。



	CUSTOMER_ID	CUSTOMER_NAME	AREA_ID	PRODUCT_ID
1	1	客户_1	21	101
2	2	客户_2	10	102
3	3	客户_3	22	101
4	4	客户_4	24	109

	PRODUCT_ID	PRODUCT_NAME
1	101	键盘
2	102	鼠标
3	103	主板
4	104	显示器
5	105	内存

图 8-19

通过更改“Join Type”，如图 8-20 所示，我们能在此视图界面快速建立诸如左外连接、右外连接等数据表连接方式。

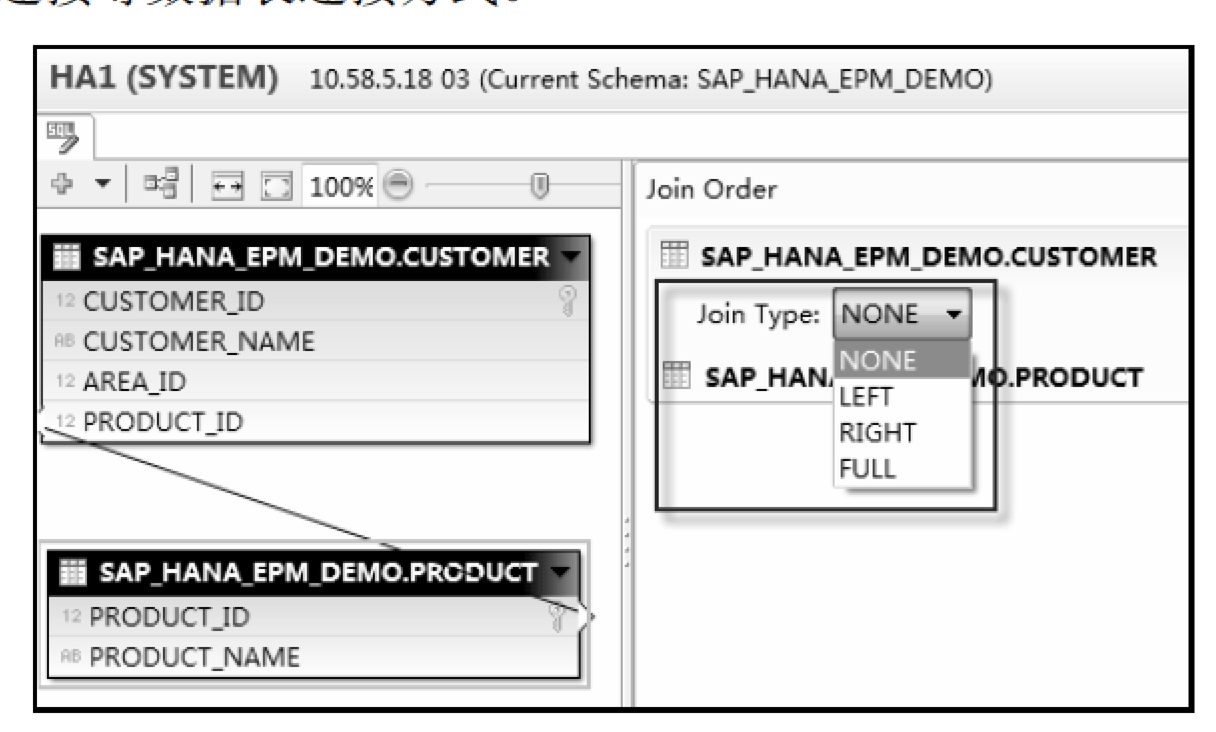


图 8-20

三、导入和导出数据表

1. 导出数据表

(1) 定位到需要导出的数据表，如图 8-21 所示。

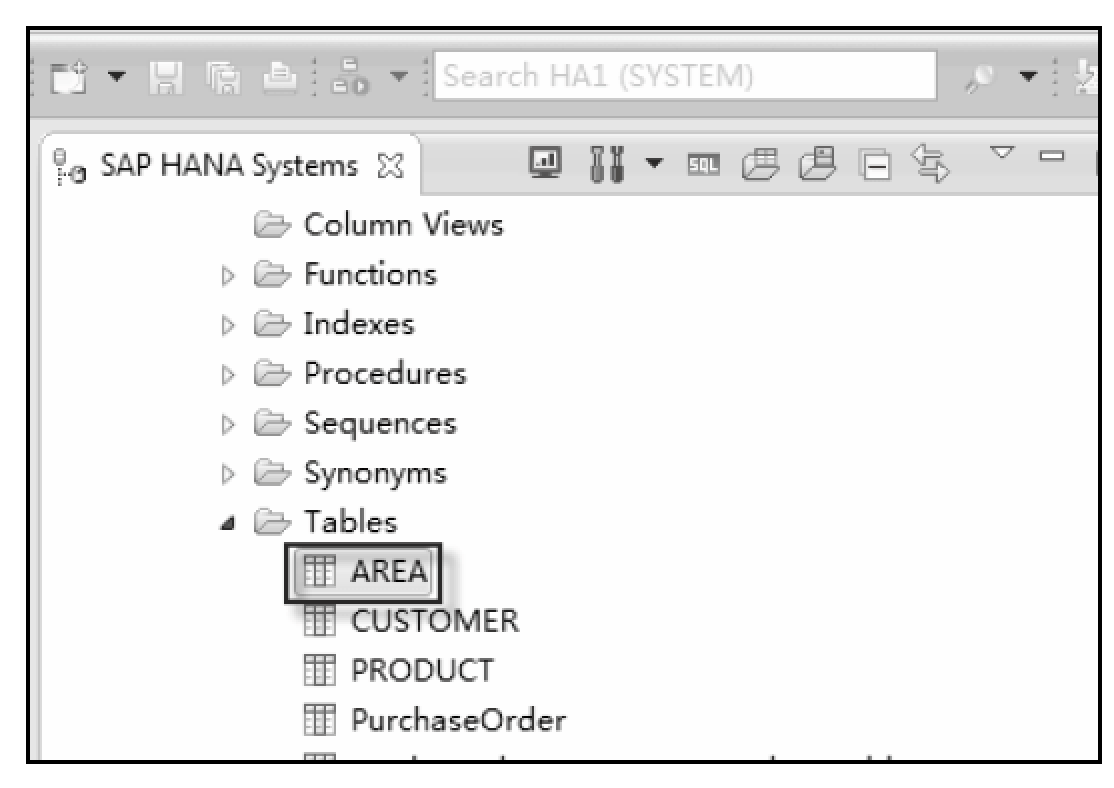


图 8-21

(2) 右击需要导出的数据表，并选择“Export”(见图 8-22)。

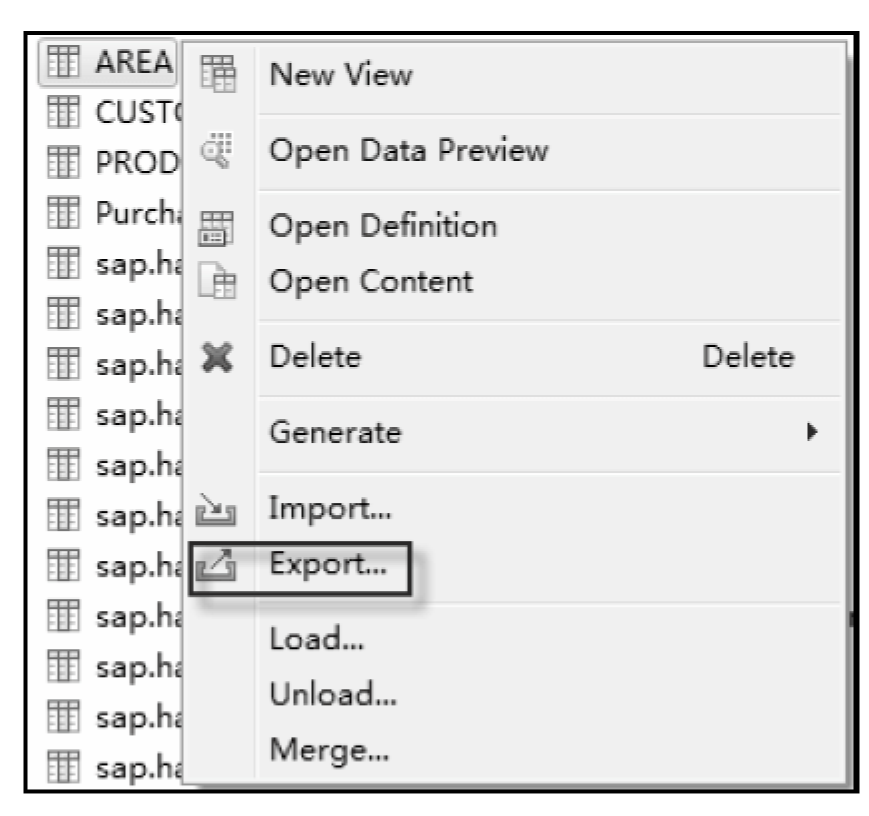


图 8-22



(3) 在输出界面，你能看到选中的数据表已经在输出区域，单击“Next”按钮继续(见图 8-23)。

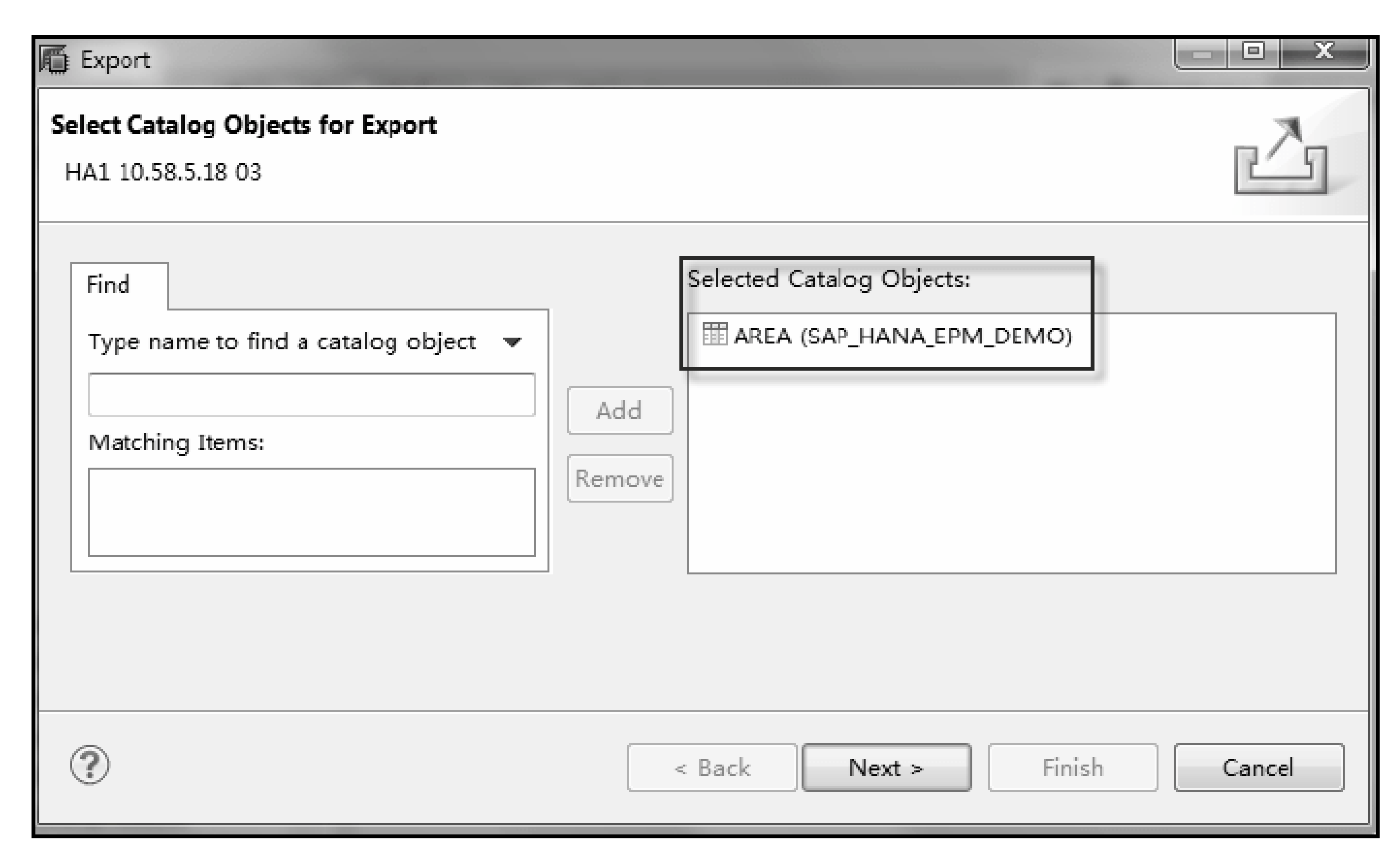


图 8-23

(4) 你可以在输出参数配置界面选择输出文件格式和输出内容(是否包含结构)，以及定义输出文件的存储位置(服务器或者本地)。全部设置完毕后单击“Finish”按钮确认(见图 8-24)。

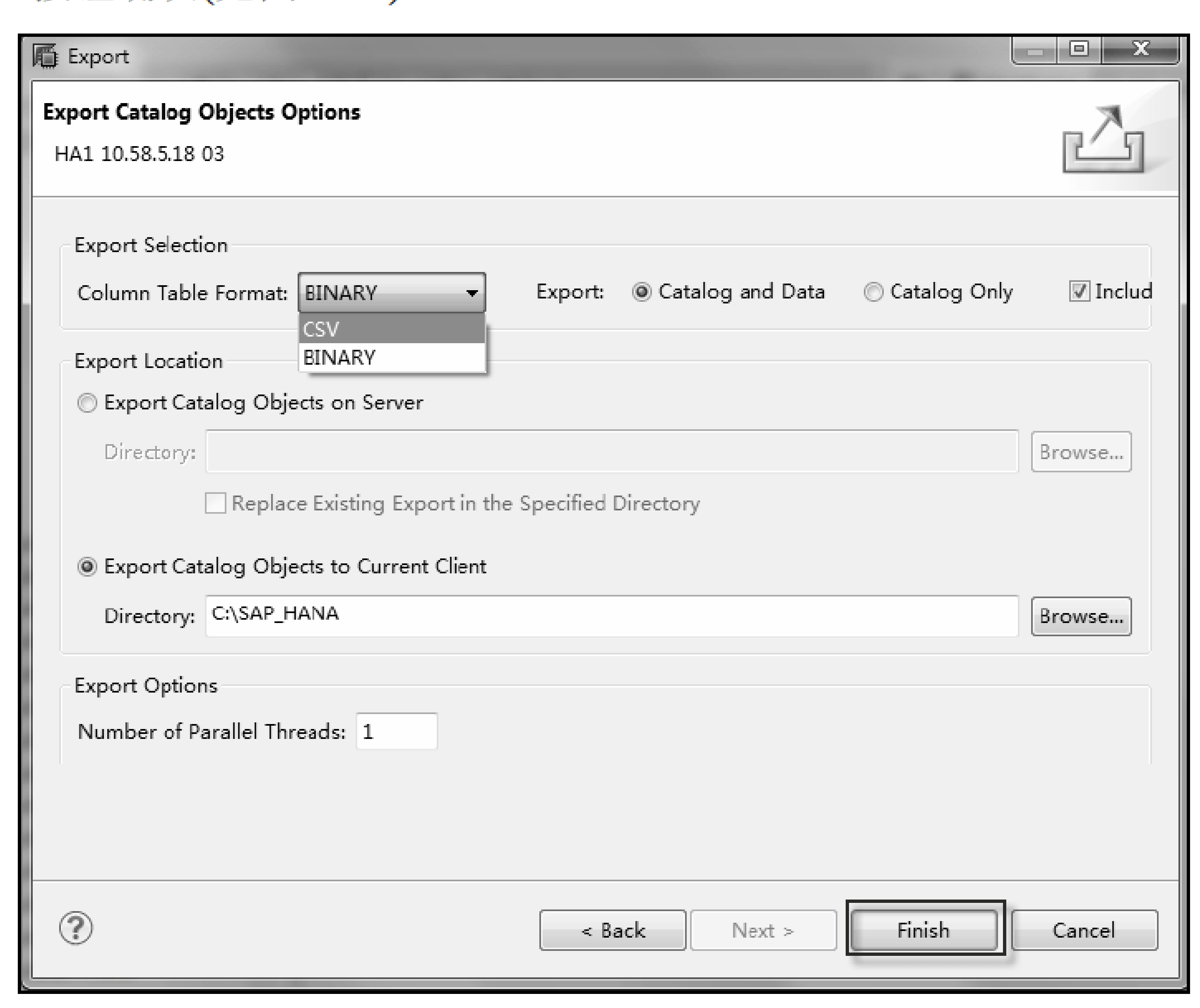


图 8-24



2. 导入数据表

(1) 定位到需要导入数据表的“Tables”文件夹，如图 8-25 所示。

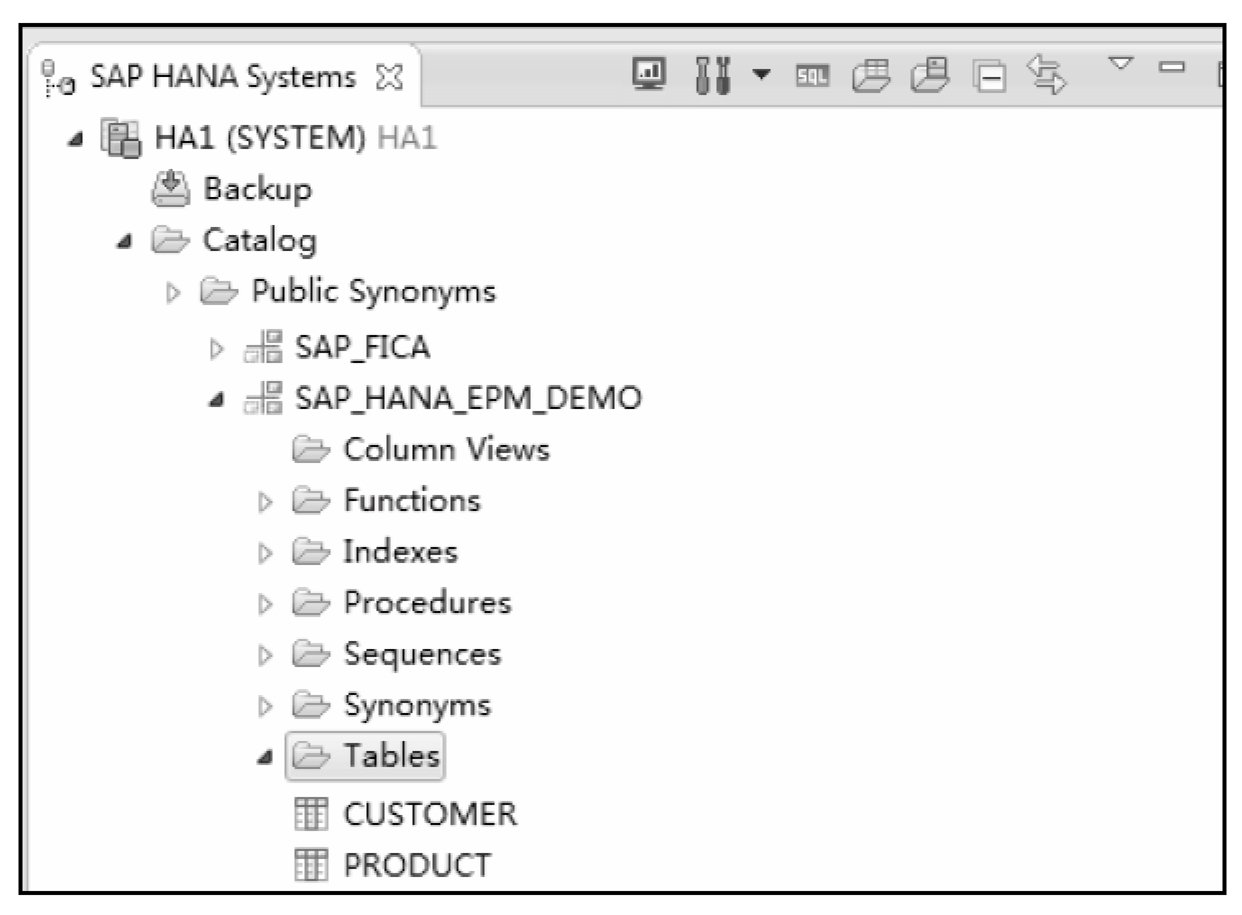


图 8-25

(2) 右击此文件夹，选择“Import”(见图 8-26)。

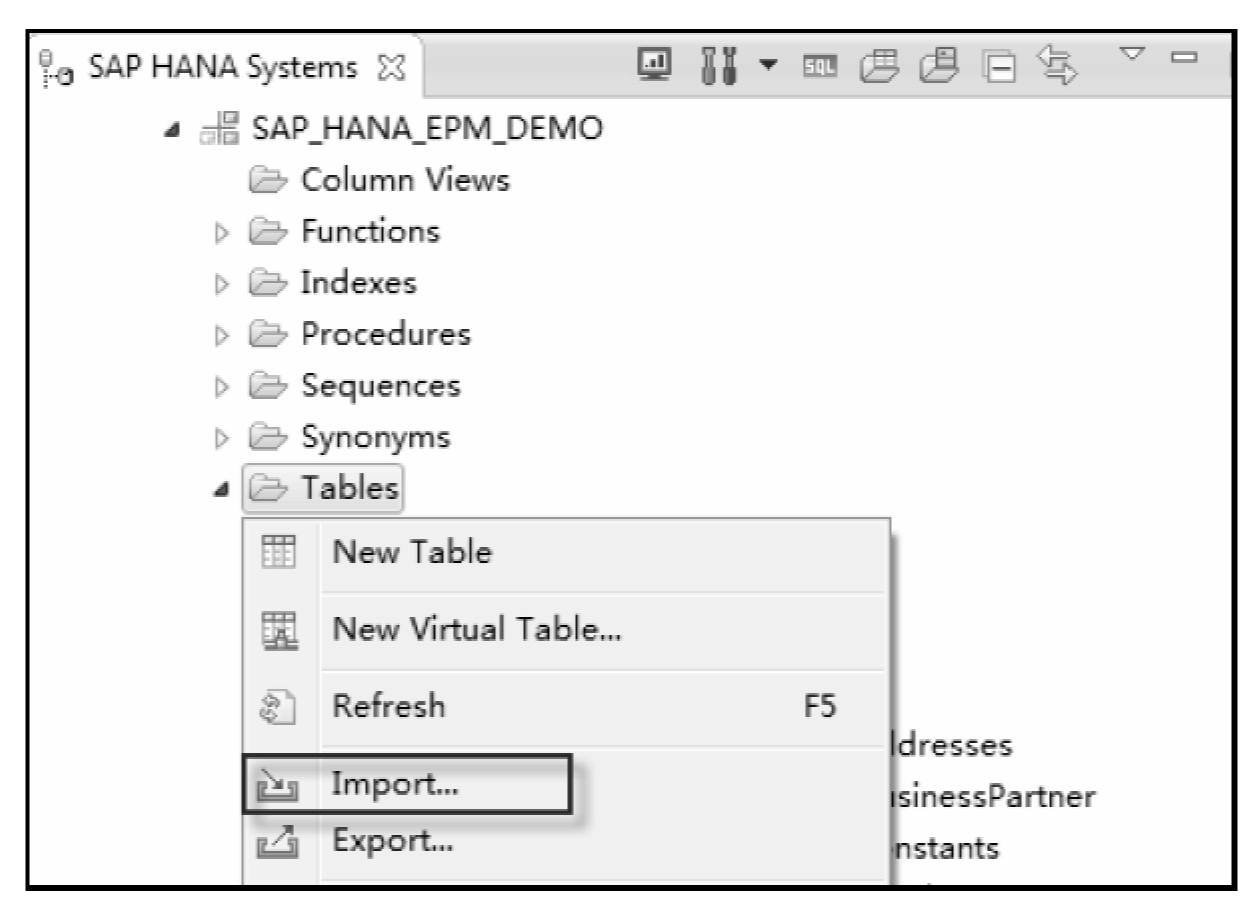


图 8-26



(3) 选择需要导入的数据表的存储位置，例如从服务器导入还是从本地导入，单击“Next”按钮(见图 8-27)。

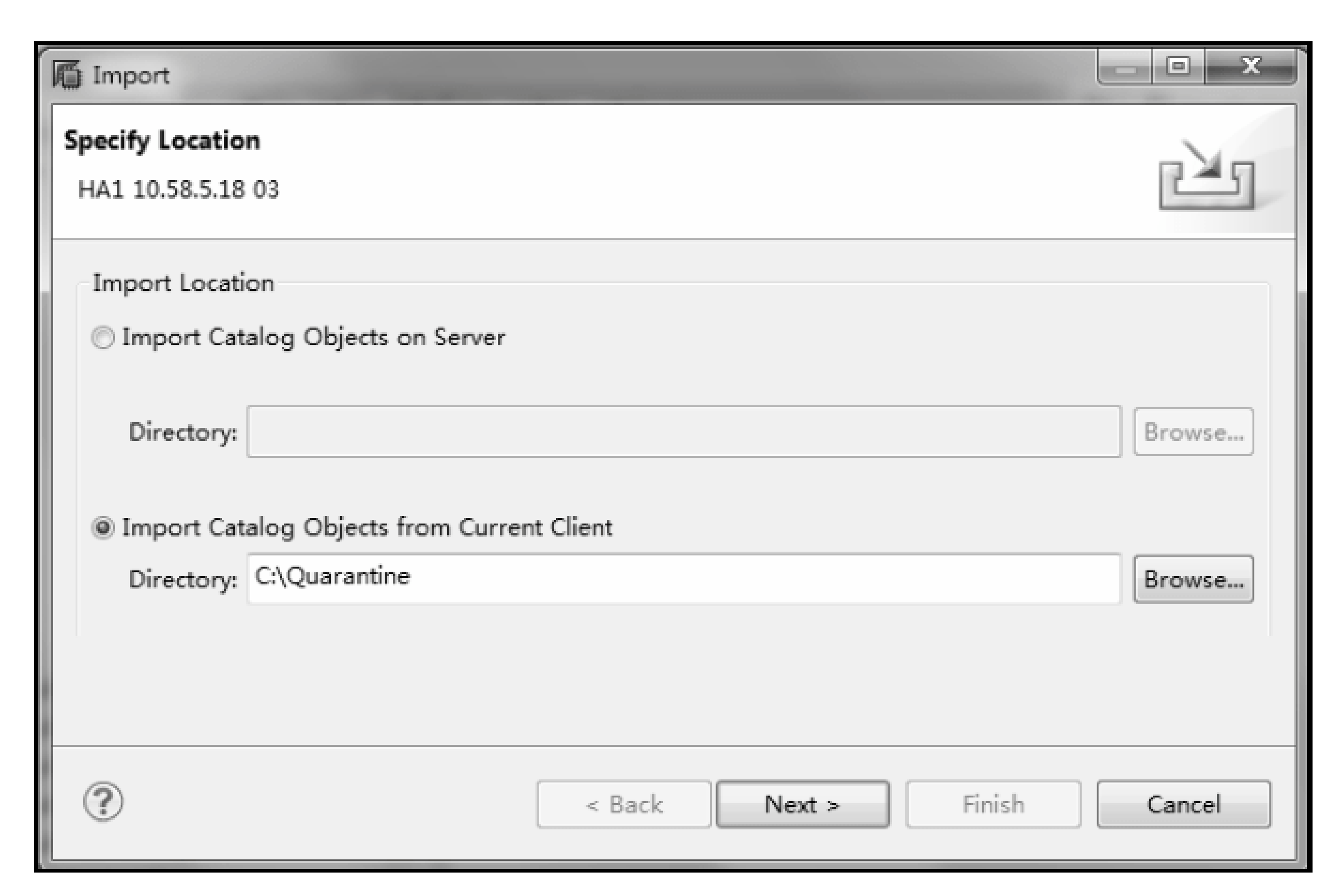


图 8-27

(4) 在图 8-28 所示的界面中选择需要导入的数据表，然后单击“Add”按钮至右侧空白区域，单击“Next”按钮。

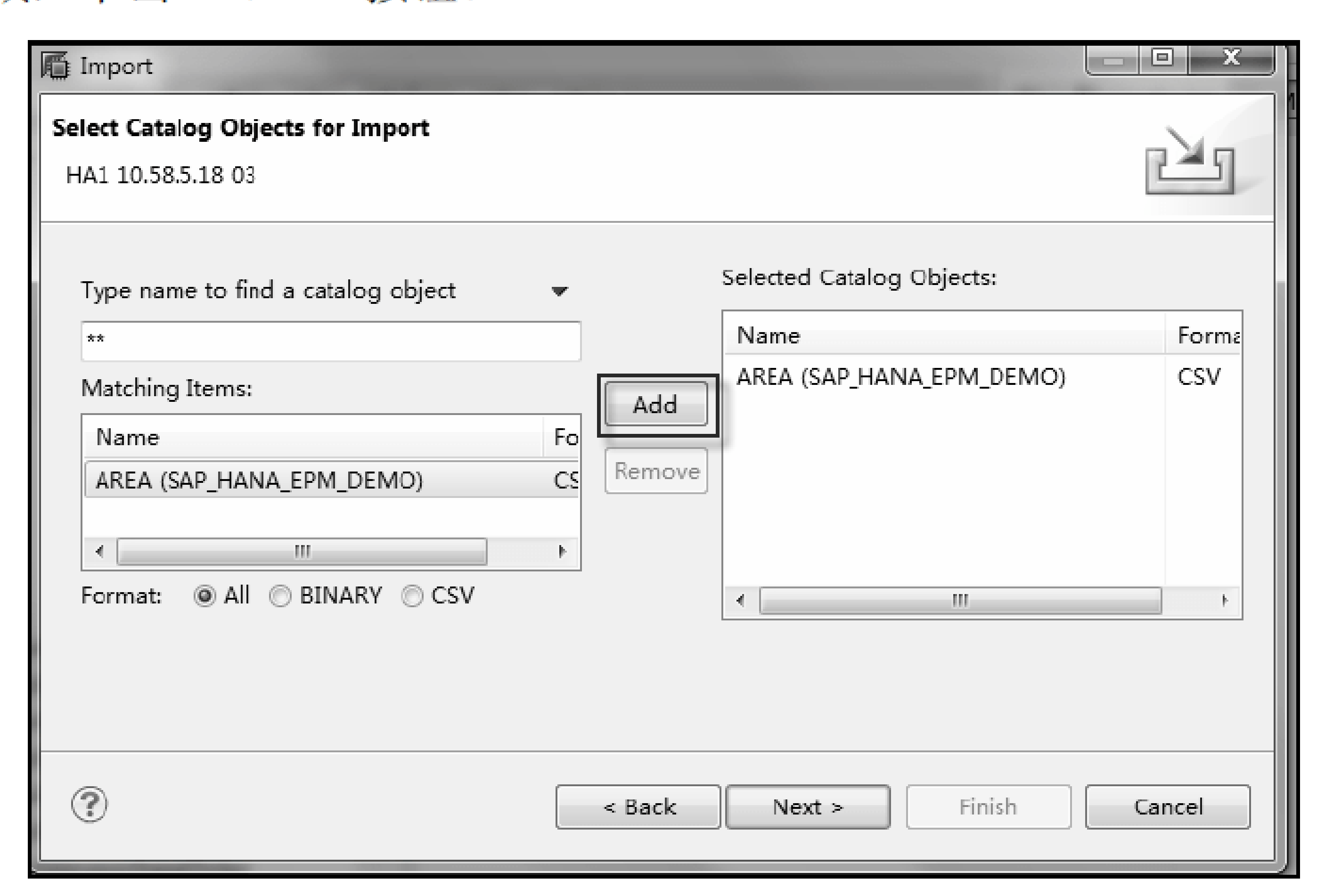


图 8-28

(5) 与数据表导出一样，在下一屏你能选择需要导入的内容(是否包含结构)以及是否覆盖已存在内容，单击“Finish”按钮确认(见图 8-29)。



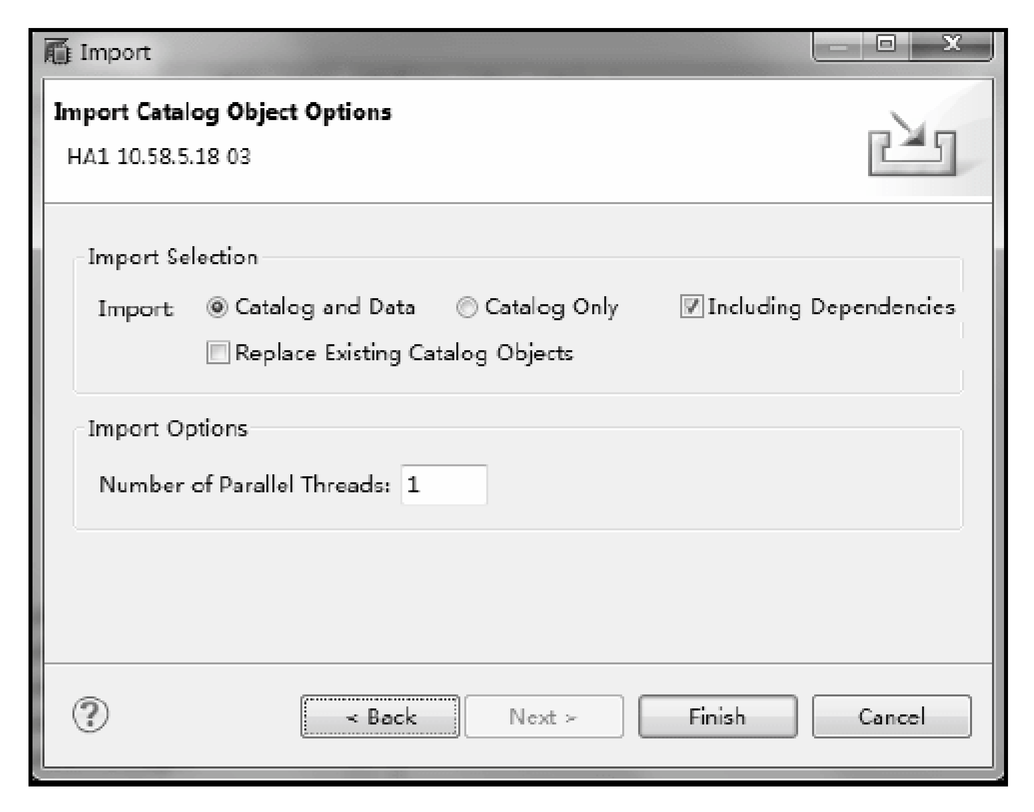


图 8-29

3. 检查数据表

SAP HANA Studio 提供了专门的存储过程(Procedure，具体描述会在第十章介绍)来检查存在于系统内的数据表是否有错误，具体调用此 Procedure 的语法为：

```
CALL CHECK_TABLE_CONSISTENCY ('<action>', '<schema_name>', '<table_name>')
```

在这里我们使用这个 Procedure 来检查刚刚导入的数据表是否有错误，在 SQL 控制台输入如下代码：

```
CALL CHECK_TABLE_CONSISTENCY ('CHECK', 'SAP_HANA_EPM_DEMO', 'AREA');
```

得到如图 8-30 所示的运行结果。



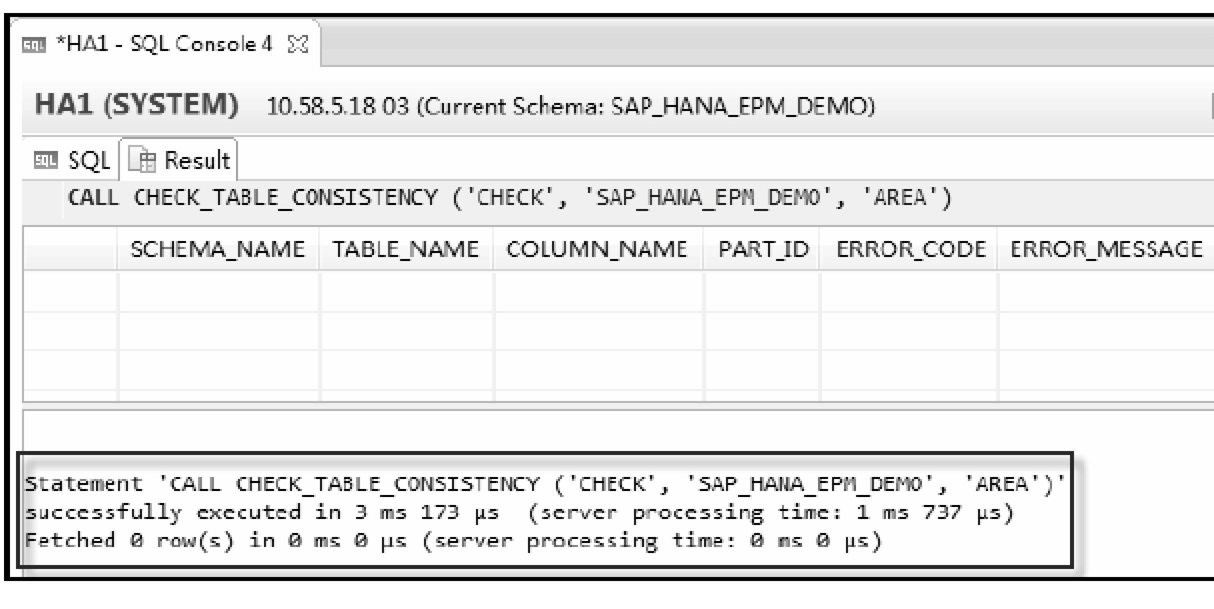


图 8-30

从返回结果看刚刚上传的数据表是没有任何错误的，可以使用。

接下来我们将介绍如何在 SAP HANA Studio 中创建信息模型。通常来讲，创建

信息模型包含以下几方面内容：

- 创建包
- 创建属性视图
- 创建分析视图
- 创建计算视图
- 创建过程
- 创建分析特权

从下一小节开始，我们将详细介绍如何创建包到创建分析特权。

第三节　SAP HANA Studio 之包简介

一、包简介

在 SAP HANA 中，包(Package)是用来包含一批 SAP HANA 仓储对象(Repository Objects)并用于不同 SAP HANA 系统之间传输使用的。多个包可以合并到一个交付单元(Delivery Unit)中用来进行整个项目的交付。在 SAP HANA 仓储(Repository)中，每个对象都会被分配给一个包，而每个包都会对应一个指定的交付单元。在讲述如何创建包前，我们先来解释几个知识点：

- 包层次(Package Hierarchy)
- 包类型(Package Type)
- 包命名规则(Package Naming Conventions)

1. 包层次

通过建立包层次，你能为多个包建立父子关联类型。这样做的好处是，你可以在这些父子关系的包中按照项目的逻辑顺序区分项目的不同部分以使层次结构化，方便以后对项目内容的查找。还是以 EPM 为例，从图 8-31 中我们可以看出不同的子包包含了不同的项目内容，如“设置”、“管理”、“数据”、“模型”等。当我们需要对项目某一部分进行更新或修改时，只需要进入对应的子包中操作即可。不仅如此，SAP HANA 的交付单元是不依赖于包层次的，即如果你只对于项目中某个子包的内容进行更新，就可以单独把这个子包放入交付单元进行传输，这样做将

会大大减少每次更新的时间和数据量。

根据 SAP 的最佳实践，我们推荐大家在新建项目时优先考虑建立包层次而不是把所有内容放在一个包中。图 8-31 是 SAP 官方的 EPM 实例的包层次：

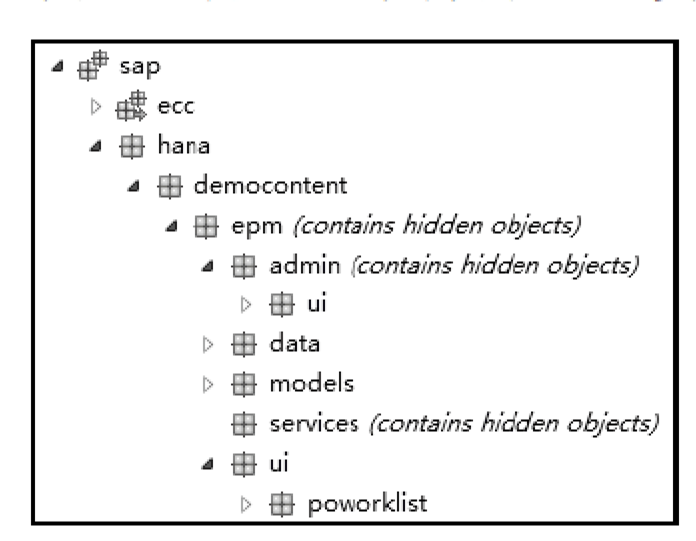


图 8-31

对于包层次，我们有一点需要着重指出的是，SAP HANA 中所有由 SAP 公司提供的内容都包含在“sap”根包下的子包里面。因此请不要在“sap”根包下创建任何与你自己项目相关的包，因为这些包及其包含的内容可能会在下次 SAP 公司对交付内容进行更新时丢失或者损坏。我们推荐你新建一个根包及其子包来包含所有项目相关的内容。

2. 包类型

在 SAP HANA 中，用户可以创建的包分为结构化和非结构化两种类型，如图 8-32 所示。

- 结构化：结构化包只包含子包，并不包含库对象(如数据模型或存储过程)。
- 非结构化：非结构化包可以同时包含子包和库对象。

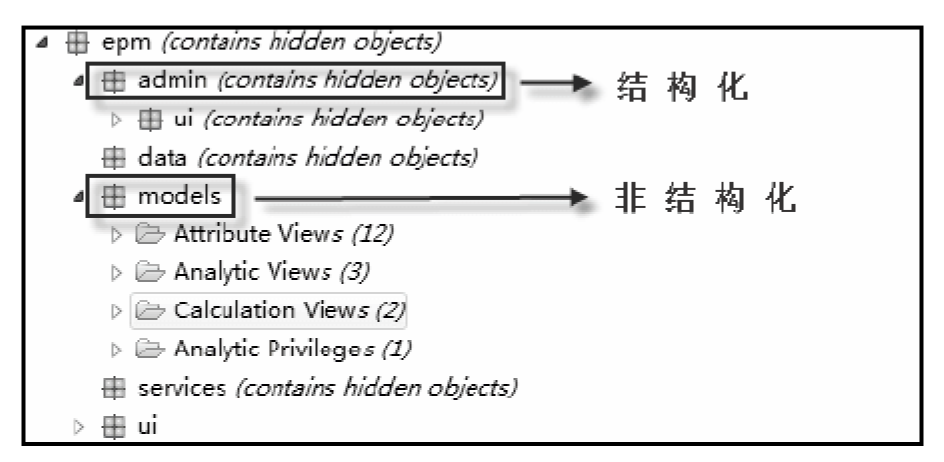


图 8-32



例如在图 8-32 中，“admin”包是结构化的，它只包含了“ui”子包；“models”包是非结构化的，它包含了很多库对象，如各种 SAP HANA 视图。

除了用户可以创建的包外，SAP HANA 还有几种自带的包：

- “sap”：SAP HANA 专门为 SAP 公司交付内容预留的可传输用的包。第三方合作伙伴和客户的交付内容需要包含在除此之外的包中，即第三方合作伙伴和客户是不可以使用“sap”包来交付内容的，具体原因我们前面已经解释过。
- “system-local”：“system-local”包及其包含的子包是不可以用来传输的。其包含的所有内容都被视为系统本地的内容并且不能传输到其他系统中。假如你熟悉 ABAP 的开发会更易理解，因为这个包和 ABAP 中的“$tmp”开发包概念类似。
- “system-local.generated”：其是非传输的结构化包，它是用来包含系统运行时所生成的本地内容的包。
- “system-local.private”：“system-local.private”包也是非传输的结构化包，它是用来包含单独的系统用户(以“system-local.private.<user_name>”子包区分)在系统中操作内容的包。为了避免日后出现系统的兼容性问题，请不要对此包及任何此包下面的子包进行任何操作。

总之，我们列出的上述包都是系统自带的包，请在任何时候都不要使用这些系统自带的包来进行项目的开发。

3. 包命名规则

当我们创建包时就需要为新建的包进行命名。在 SAP HANA 中，包的名字不能随便起，包的名字要符合特定的命名规则才可以。具体的规则如下：

- 被允许的字符：包括大/小写的(Aa-Zz)26 个字母、0～9 的数字、连字符(-)、小数点(.)。这些字符是在命名时允许的，其他特殊字符如(#)、(%)、(&)等是不可以出现在包名称中的。在为包命名时，请尽量指出该包的逻辑层次。例如我们要新建一个包“c”，这个包是包“b”的子包，同时包“b”又是包“a”的子包，那么我们命名包“c”时应该将包“c”取名为“a.b.c”而不是直接命名为“c”。
- 非法字符：包名不可以以小数点(.)或者连字符(-)开头，包名中不可以出现连续的两个或者多个小数点(..)。
- 包名称长度：包名称最多包含 190 个字符，命名请勿超过此长度。

- 包命名空间长度：完整的包命名空间，例如“aa.bb.cc.zz”(包含小数点)的最大长度也为 190 字符。

二、创建包

在本小节中，我们将介绍如何在 SAP HANA Studio 中创建包。

(1) 打开 SAP HANA Studio，切换到“Modeler”透视图(见图 8-33)。

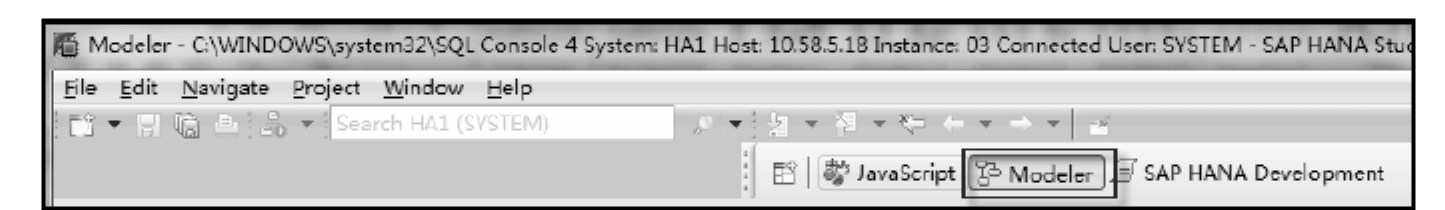


图 8-33

(2) 在“SAP HANA Systems”视图中，右击“Content”根目录调出右键菜单并选择“New”→“Package”(见图 8-34)。

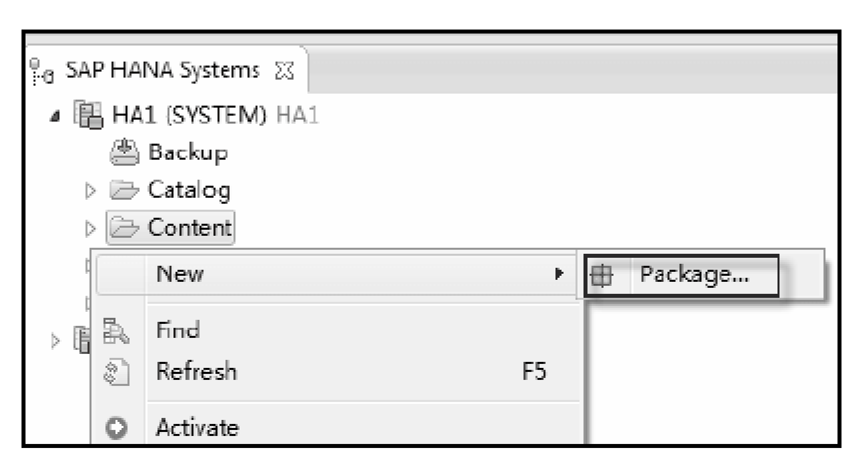


图 8-34



(3) 在“New Package”窗口中输入包的名称和描述，选择交付单元以及语言和负责人。单击“OK”按钮(见图 8-35)。

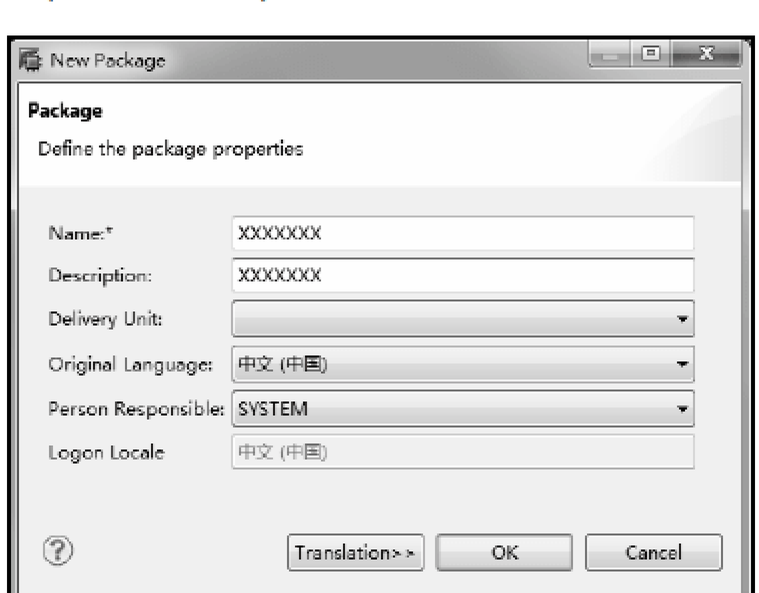


图 8-35

- 包名称(Name)是必输项。
- 交付单元(Delivery Unit)默认为空，请在下拉菜单中为新生成的包选择合适的交付单元。
- 原始语言(Original Language)定义了包的开发语言，请确保其与你登录的语言一致。
- 负责人(Person Responsible)定义了该包由哪个用户负责，默认为登录用户。

(4) 在“SAP HANA Systems”视图的“Content”根目录下右击新生成的包，例如下图的“CUSTOMER”包，右击调出菜单，单击选择“Active”激活新生成的包(见图 8-36)。

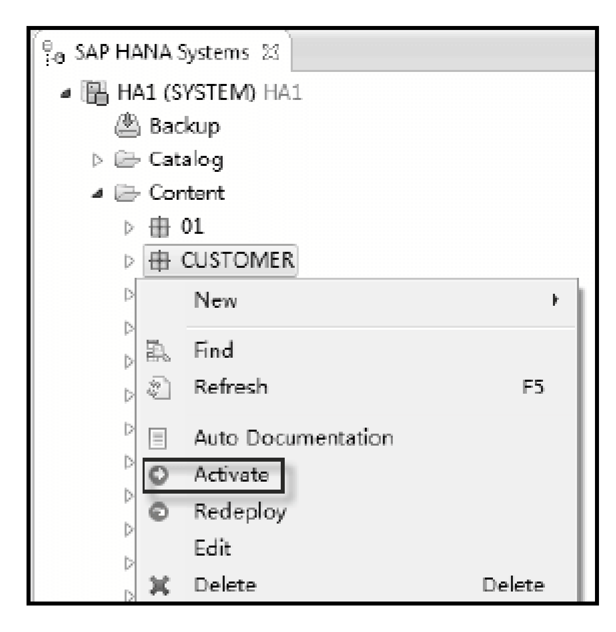


图 8-36

(5) 激活成功后新包就能使用了。

有一点需要指出的是，当我们在新建的包下面继续创建子包时，系统会默认将父包的命名空间带入子包的名称中，请在创建子包时务必不要去除系统指定的命名空间，如图 8-37 所示。

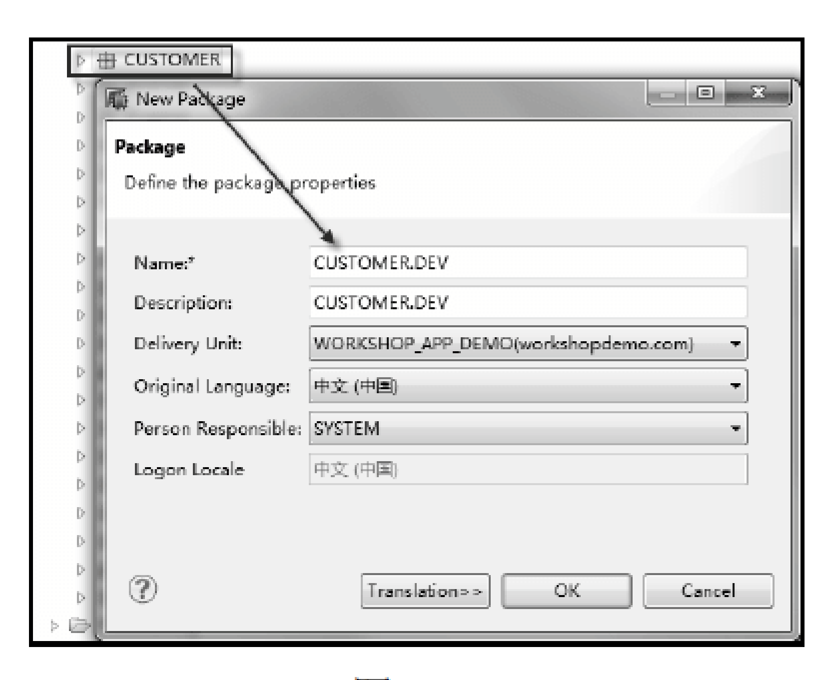


图 8-37

例如我们要创建“DEV”子包于“CUSTOMER”包下，则包名称应为“CUSTOMER.DEV”。

三、定义包特权

在本小节中，我们将介绍如何定义包特权。包特权是用来限制用户在包中的操作权限的，简单来说包特权就像一把钥匙，有了这把钥匙你就能进入包并对其包含内容进行读取或者修改，没有钥匙的话，你就无法进入这个包中(当你想打开包中的内容时会被提示没有权限)。特别要指出的是，当你拥有根包的特权时，你就同时拥有了对其下包含的子包同样的特权。

为包建立特权的操作如下。

(1) 打开 SAP HANA Studio，切换到“Modeler”透视图(见图 8-38)。

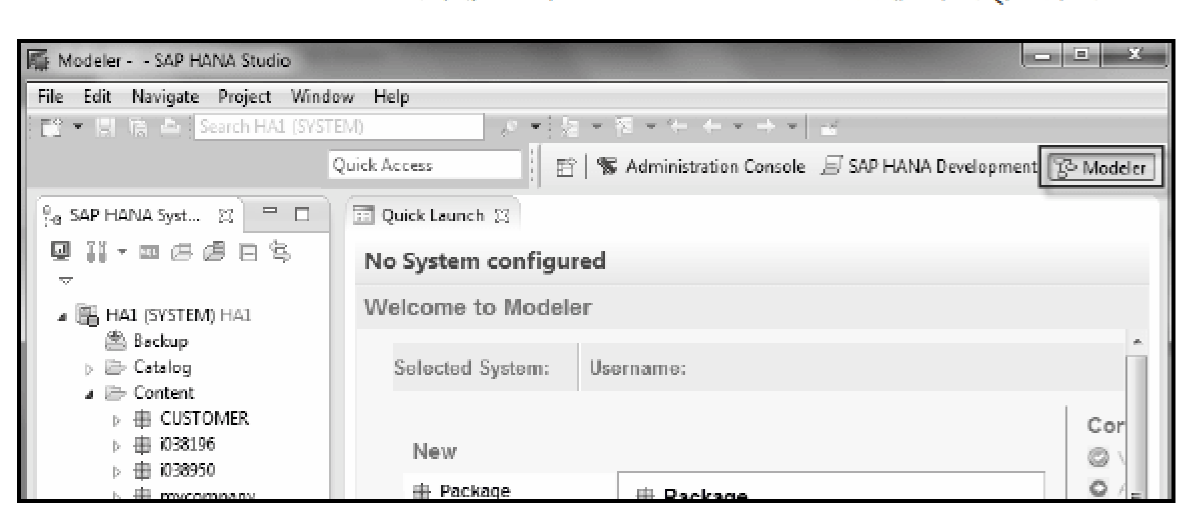


图 8-38

(2) 在“SAP HANA Systems”视图中定位到“security”→“Users/Roles”(见图 8-39)。

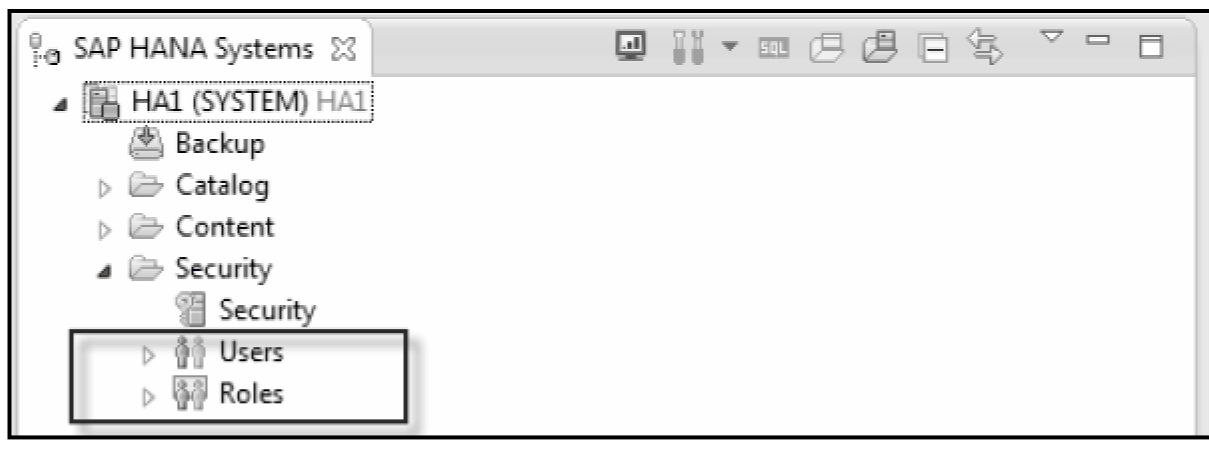


图 8-39



根据你是要分配给用户还是角色包特权，你可以展开“Users”或者“Roles”节点来选择你要分配的用户/角色。本例中我们给“SAP_EPM_USER”这个用户来分配包特权。

(3) 双击你要分配包特权的用户/角色(本例中为“SAP_EPM_USER”)，并定位

到“Package Privileges”标签页(见图 8-40)。

图 8-40

(4) 单击“+”按钮，在“Select Repository Package”窗口选择需要添加特权的包，单击“OK”按钮(见图 8-41)。

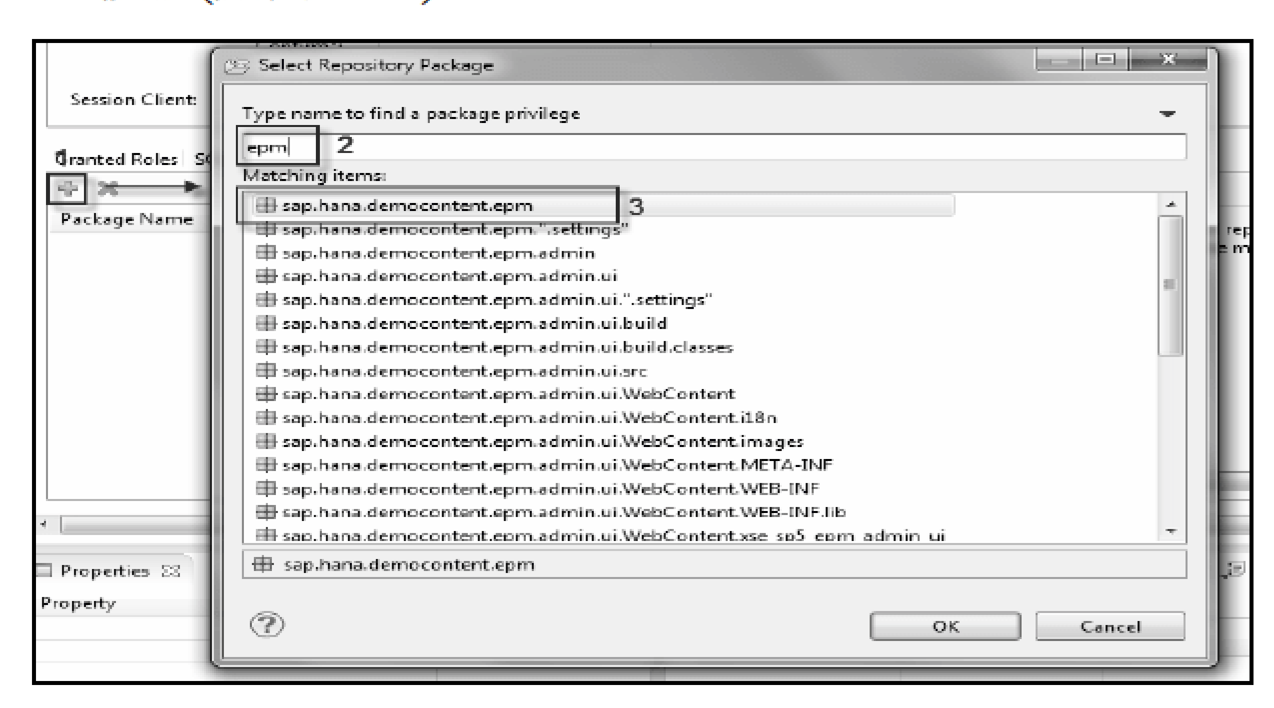


图 8-41

需要说明的是，“Select Repository Package”窗口初始打开时是不提供任何选择项的，你需要在图 8-41 中标识为“2”的查询字段输入包名称中的全部或部分字符并按回车键，系统会自动搜索出所有与之匹配的包。另外还有一点是，正如我们前面介绍的，如果你把父包对某一用户/角色授权，则其包含的子包会自动授权给该用户/角色。例如我们想让“SAP_EPM_USER”用户能访问所有 EPM 包的权限的话，只需要将 EMP 根包“sap.hana.democontent.epm”的特权赋予该用户即可。如果你只想让“SAP_EPM_USER”用户访问 EMP 根包下几个特定子包，则需要将此根包下的子包分别授权给该用户。

(5) 新分配的包指定特权。选中新分配的包后，你能在其右边为用户/角色指定

详细的特权项目(见图 8-42)。

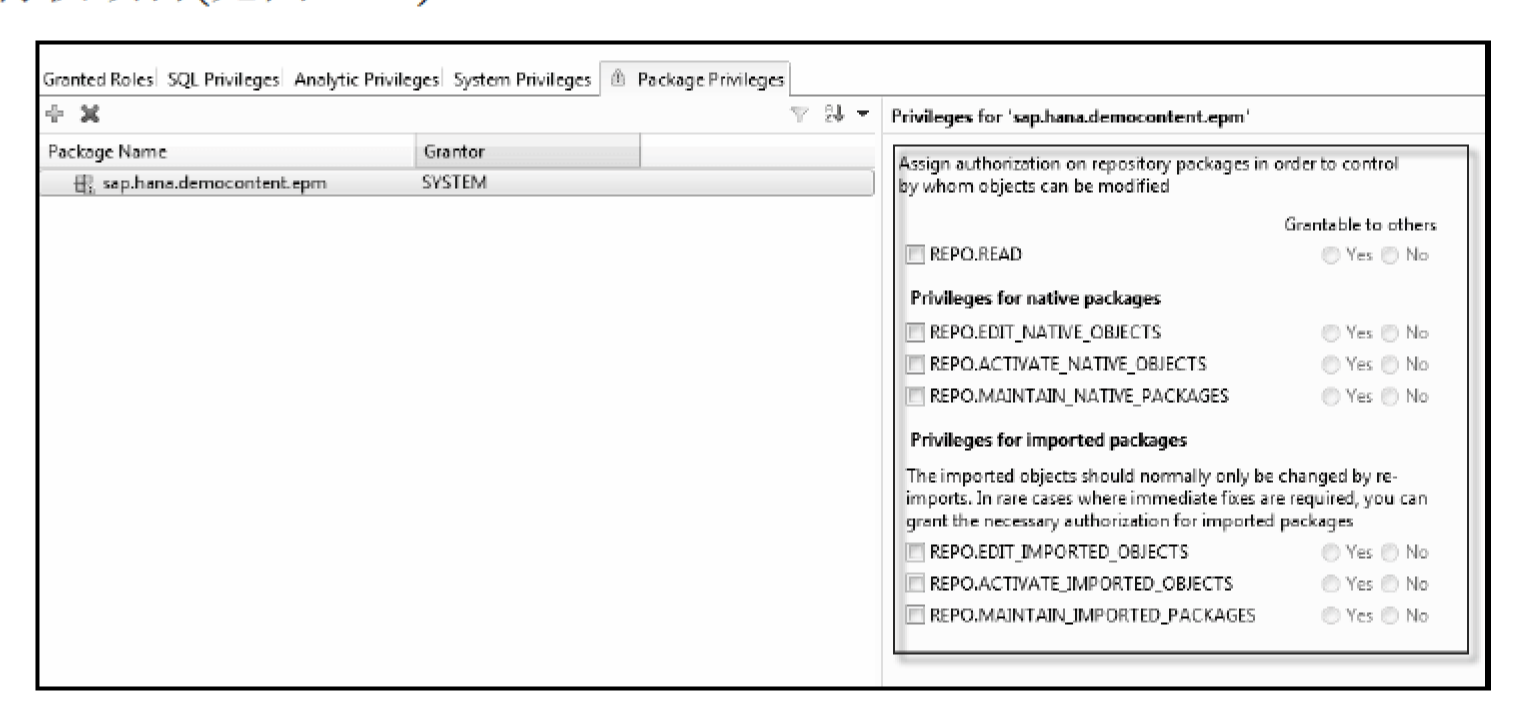


图 8-42

- REPO.READ：对选定包中的设计阶段对象(design-time objects)具有读取权限(对原生或者导入的包都有效)。
- REPO.EDIT_NATIVE_OBJECTS：对用户所授权的原生包中的设计阶段对象具有编辑权限。
- REPO.ACTIVATE_NATIVE_OBJECTS：对用户所授权的原生包中的设计阶段对象具有激活/重新激活权限。
- REPO.MAINTAIN_NATIVE_PACKAGES：对用户所授权的原生包具有更新/删除，以及在其下建立子包的权限。
- REPO.EDIT_IMPORTED_OBJECTS：对于导入的包中的对象具有编辑权限。
- REPO.ACTIVATE_IMPORTED_OBJECTS：对于导入的包中的对象具有激活/重新激活权限。
- REPO.MAINTAIN_IMPORTED_OBJECTS：对于导入的包具有更新-删除以及在其下建立子包的权限。



第四节　创建属性视图

我们在前面讲过，属性视图的作用就是把不同的关于描述性属性的表连接起来变成一个单一维度的主数据表。例如我们可以把“客户名称”、“客户代码”、

“客户地址”、“客户电话信息”等各种属性字段从它们所属的不同的数据表中提取出来并连接成一个关于客户整体信息的主数据视图。

通过使用属性视图，我们可以在被连接的表中选择哪些列和行是我们组成主数据视图所需要的，并且还可以为选择的行加筛选条件。对于属性来讲，对只具有描述性的属性数据字段使用类似于求和或求平均之类的计算是没有意义的，所以我们不需要在属性视图中定义度量和聚合数据。由于属性视图可以被视为一个单一维度的主数据表，因此你可以将属性视图与其他属性视图或者数据表在分析视图中做连接操作。

在 SAP HANA 中，属性视图是可以在多个对象中共享的，即同一个属性视图可以同时分配给不同的分析视图或计算视图，并且当属性视图有任何更改时，与之相关的分析视图或计算视图会自动接受这些变更。

下面我们来看看创建属性视图的步骤：

(1) 打开 SAP HANA Studio，切换到“Modeler”透视图(见图 8-43)。

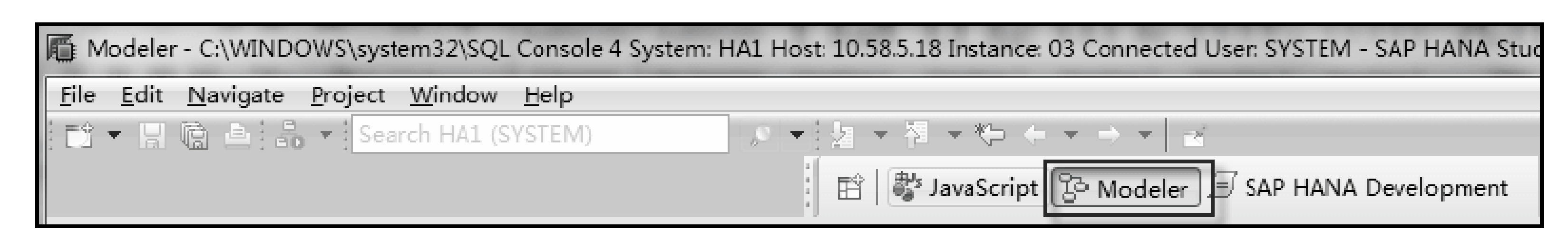


图 8-43

(2) 在“Quick Launch”视图中单击“Attribute View”选项，然后单击“Create...”按钮(见图 8-44)。

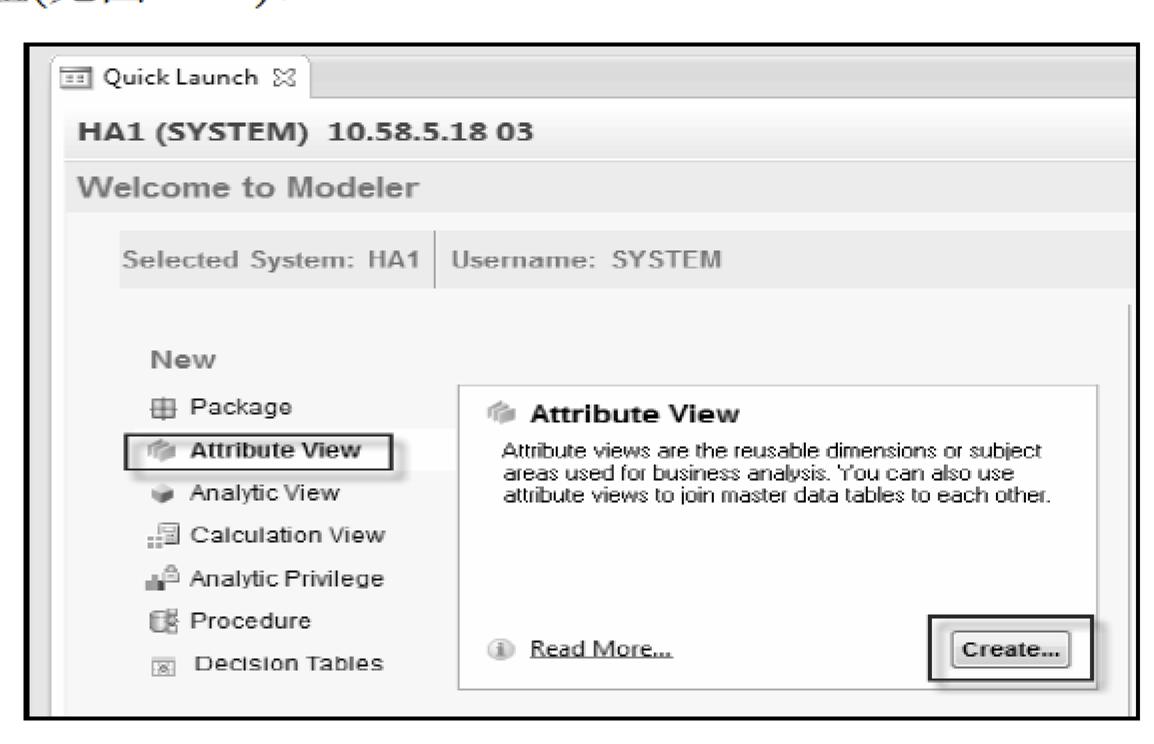


图 8-44

(3) 你还可以通过在“SAP HANA Systems”视图定位你需要创建视图的包，右



击调出右键菜单，选择“New”→“Attribute View”。本例中我们选择的包是“epm”下的“models”子包(见图 8-45)。

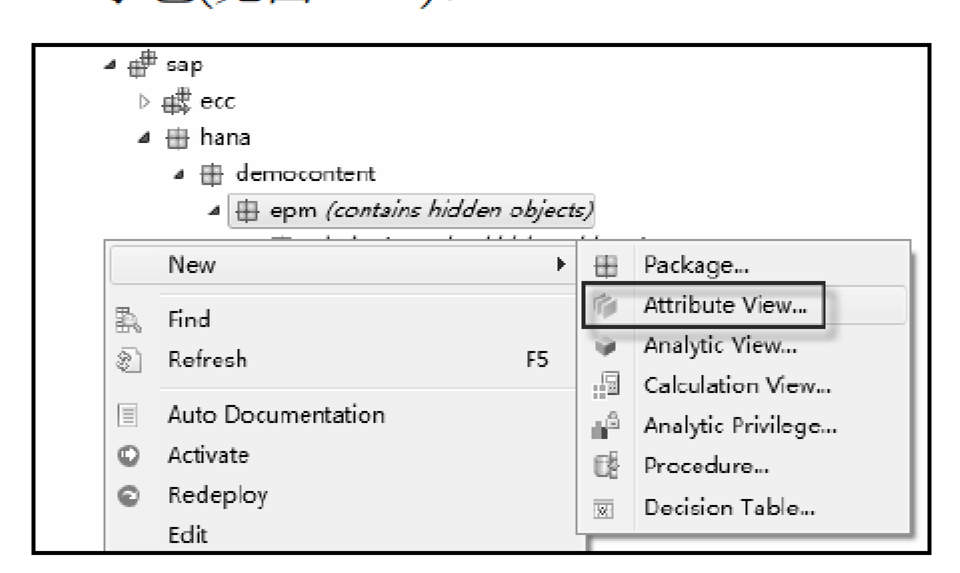


图 8-45

(4) 在“New Information View”窗口中把将属性视图的名称和描述填好。其他字段解释如图 8-46 所示。

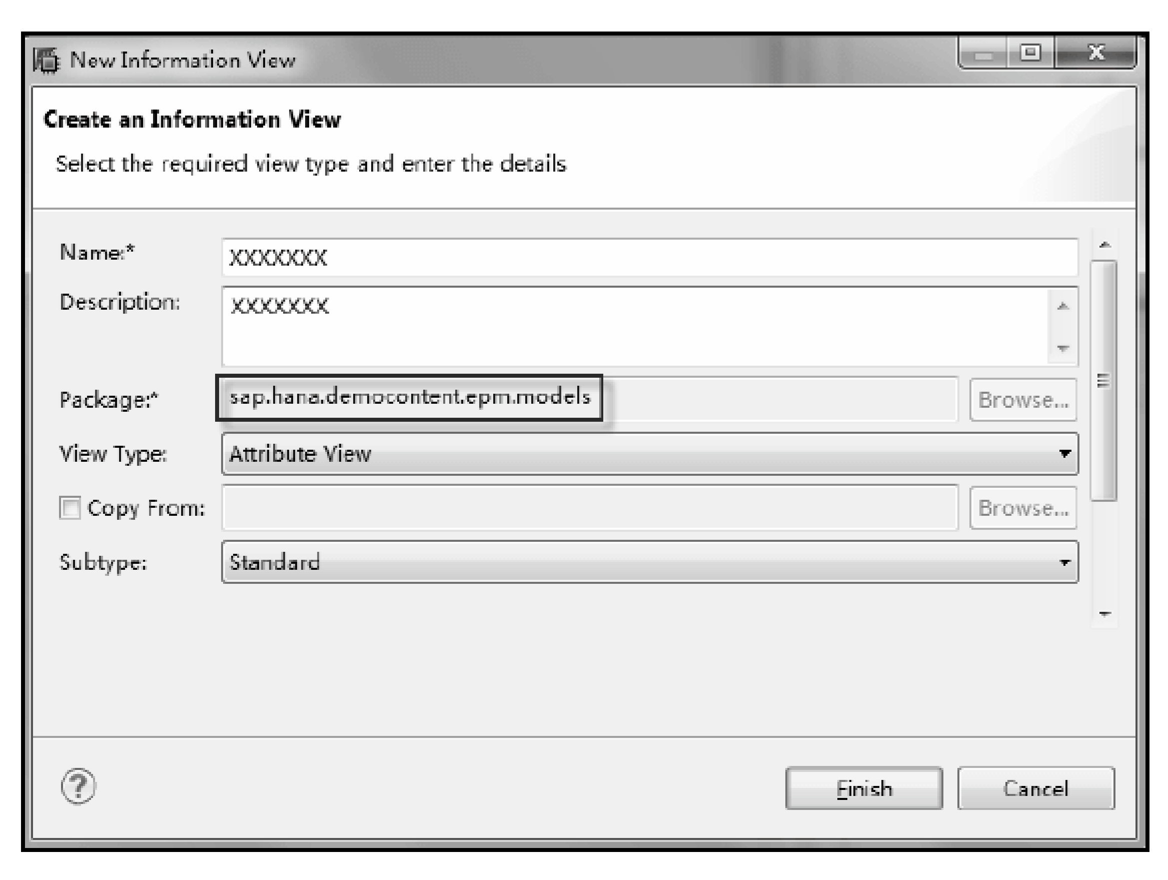


图 8-46

- “Package”包是必填项，如果你通过第 2 步创建属性视图，则需要为属性视图指定所属的包，如果你通过第 3 步创建属性视图，则系统会自动将包信息带出来。
- “View Type”选择“Attribute View”。由于本窗口还能创建其他视图，因此你需要在此指定视图类型为属性视图才行。
- “Copy From”能够提供视图复制功能。你可以复制一个已经存在的视图为新视图的基础。



- “Subtype”分为三种类型，即“Standard”、“Time”和“Derived”。如果你是基于数据表制作属性视图，则需要选择“Standard”类型。如果你需要生成时间属性视图，则需要选择“Time”类型。“Time”类型属性视图常用于具有时间特征的主数据表，例如公历的年、月、星期或者公司的财政日历等。当我们把“Subtype”切换成“Time”后(如下图所示)，系统会自动出现相关的时间选项供你配置。如果你选择公历(Gregorian)，则需要在时间间隔(Granularity)下拉表中选择合适的时间间隔。“Auto Create”选项是默认勾选的，即系统会根据你的配置自动生成时间主数据视图(见图 8-47)。由于财政日历(Fiscal)会涉及更复杂的操作，我们在这里暂不介绍。

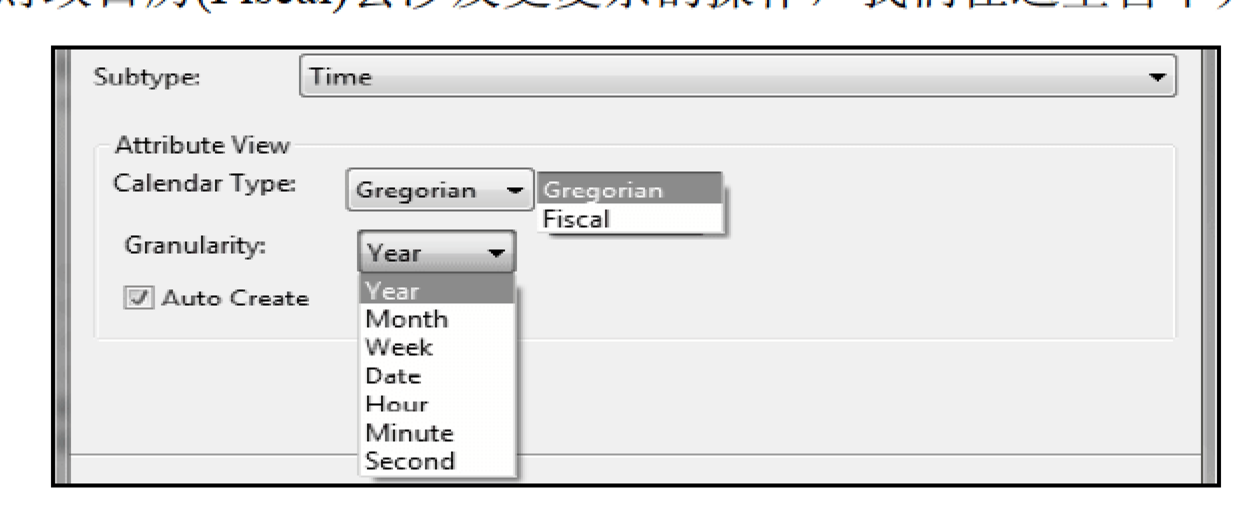


图 8-47

“Derived”类型是用来定义你新建的属性视图是否是其他属性视图的派生视图。我们知道属性视图可以看做单一维度的主数据表，并可以与其他数据表做连接。如果在分析视图或者计算视图中，有两张不同的表需要同时对同一个属性视图做不同类型的连接，如表 1 需要内连接，表 2 需要左连接，那么为了满足这种要求，就产生了派生视图的概念。派生视图具有和源属性视图一样的内容(可以看做源属性视图 1∶1 的复制)，但是我们不能对其内容做任何编辑，派生视图的内容只能随源视图的内容变化而变化。如图 8-48 所示，当你选择“Derived”子类型时，系统会要求你指定该派生视图的源视图。



图 8-48

(5) 配置好全部内容后，单击“Finish”按钮调出视图编辑窗口。完整的视图编

辑窗口如图 8-49 所示。

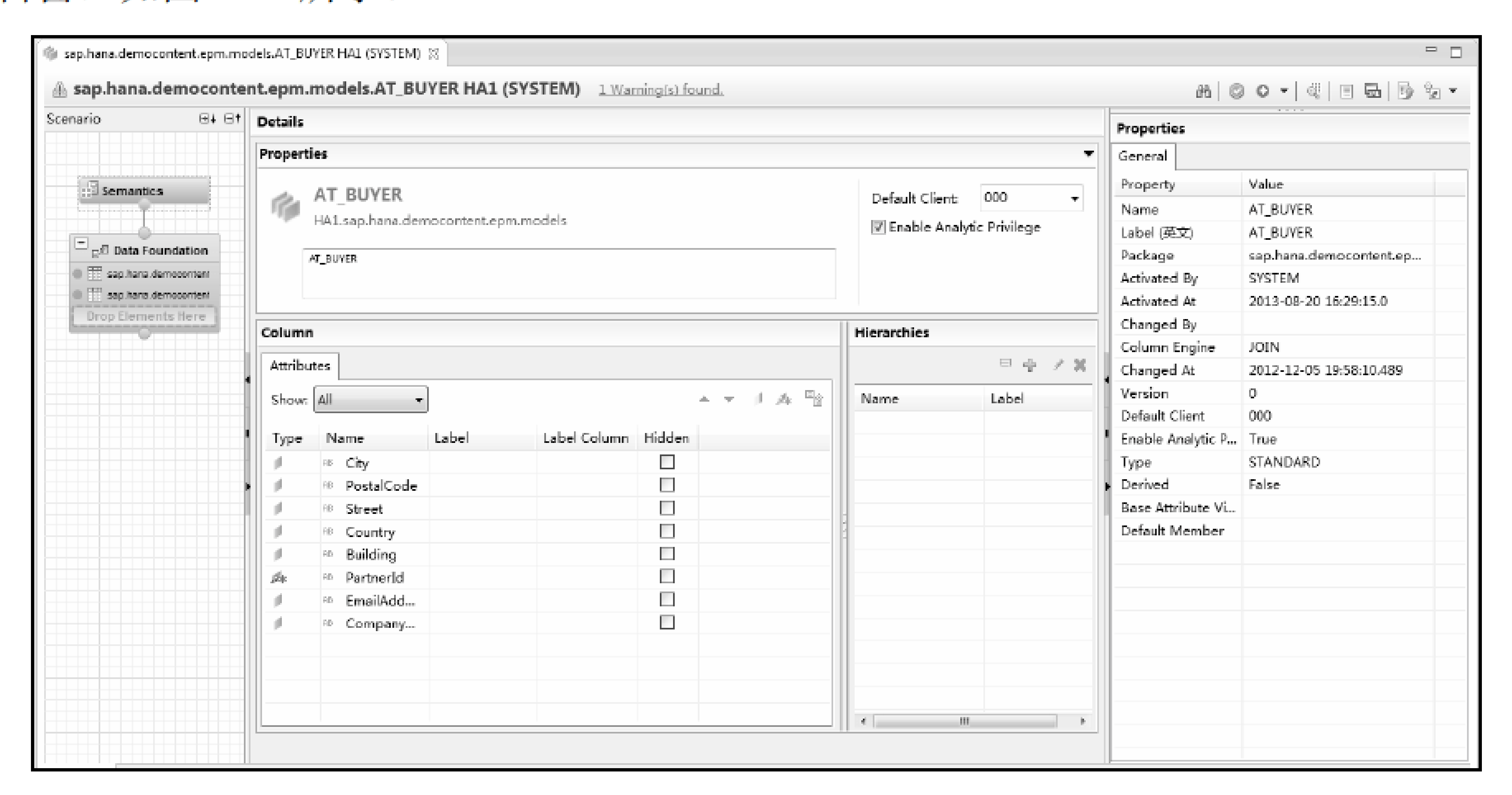


图 8-49

(6) 单击“Data Foundation”旁的“+”按钮，将需要的数据表从“Find”窗口中找到并单击“OK”按钮确认(见图 8-50)。

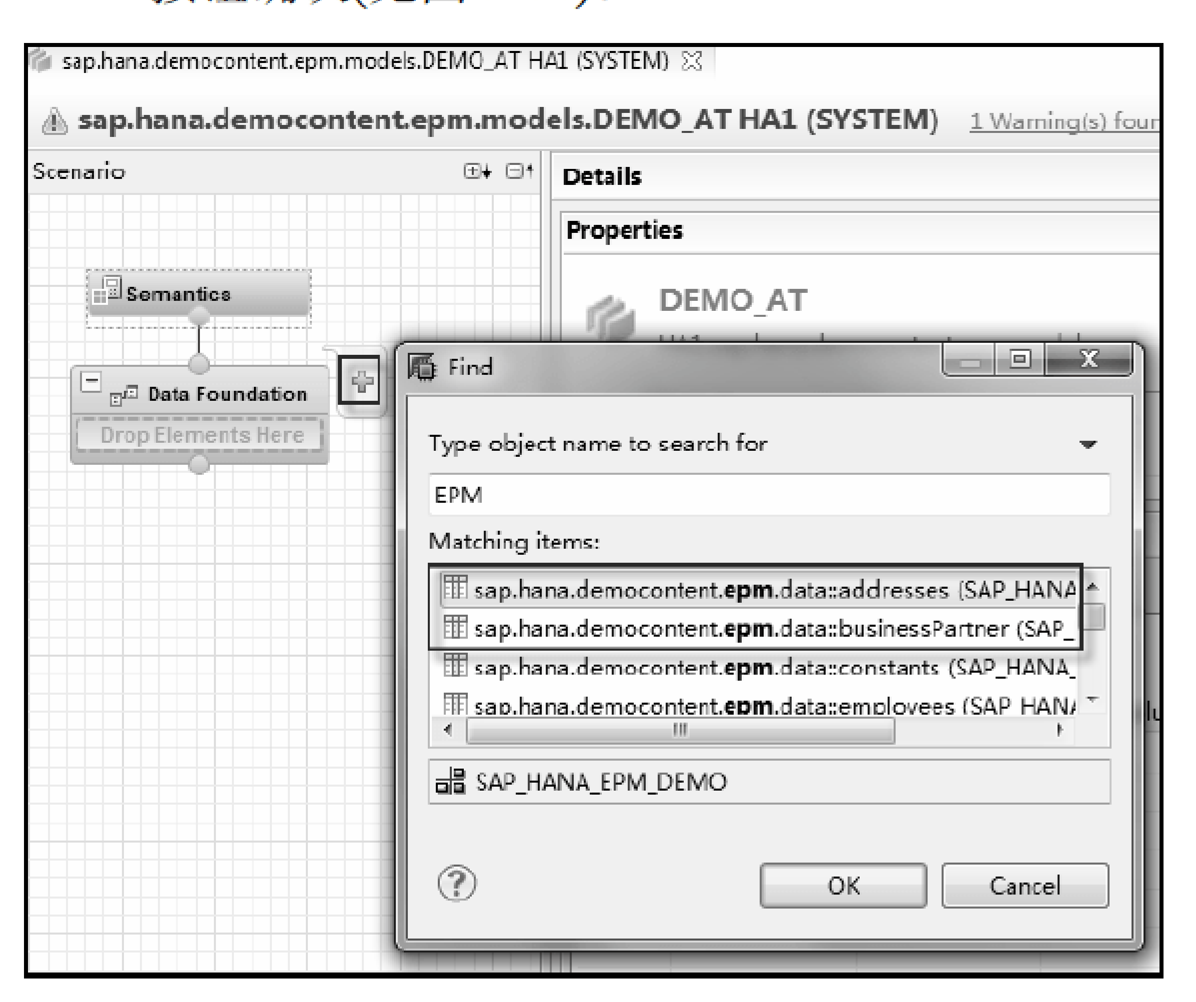


图 8-50

本例中我们将以下两张数据表“sap.hana.democontent.epm.data::address”以及

“sap.hana.democontent.epm.data::businessPartner”作为源数据表。我们选择这两个数据表的原因有，其一这两个表的相关字段组合在一起可以描述买家信息，因此能组合成一个主数据属性视图；其二为这两张表都是由属性数据组成，不涉及计算，因此非常适合作为属性视图的源数据表。

需要指出的是，我们是不能将列视图(Column Views)加入“Data Foundation”节点中的，但是我们可以将同一个数据表多次拖入“Data Foundation”节点中作为源数据表来满足我们的业务需要，我们需要做的就是为这些相同的数据表分别起别名。

(7) 创建连接。现在我们根据关键字段将上述表格连接起来，这样我们就能得到一个完整的单维度主数据表。本例中我们将“sap.hana.democontent. epm.data::businessPartner”作为主表，其他的表格作为副表，连接关键字是“AddressId”，连接方式是左外连接(请参阅前面表连接小节)。如图 8-51 所示。

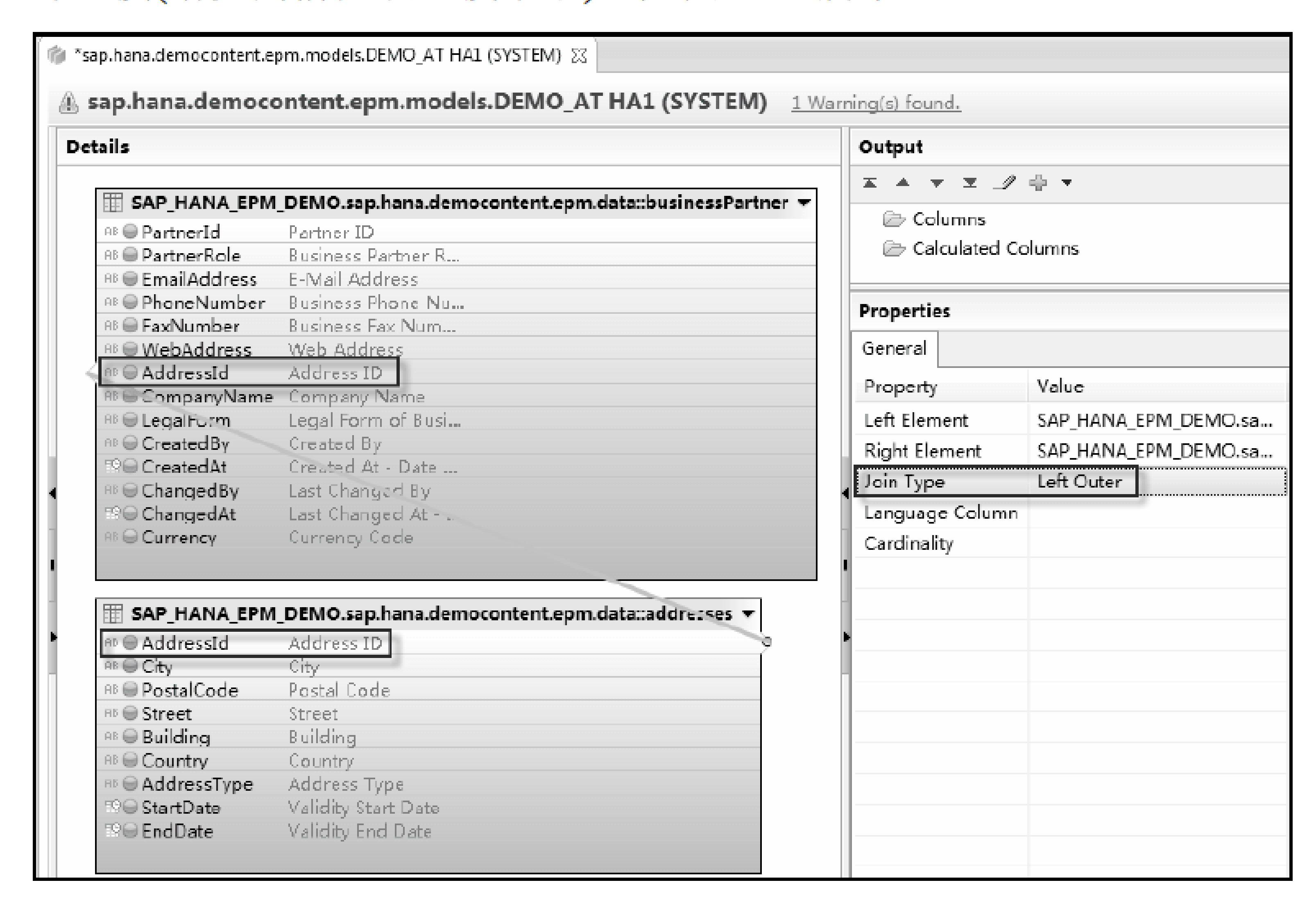
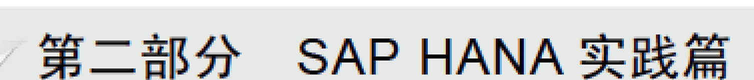


图 8-51

(8) 定义需要输出的字段。由于我们不需要将所有“Data Foundation”节点中引用的表的字段全部显示出来，因此我们需要在“Output”面板中定义需要输出的字段有哪些。定义输出字段的操作非常简单，只需要在数据表相应的字段上面右击调出右键菜单，选择“Add To Output”一项即可(见图 8-52)。

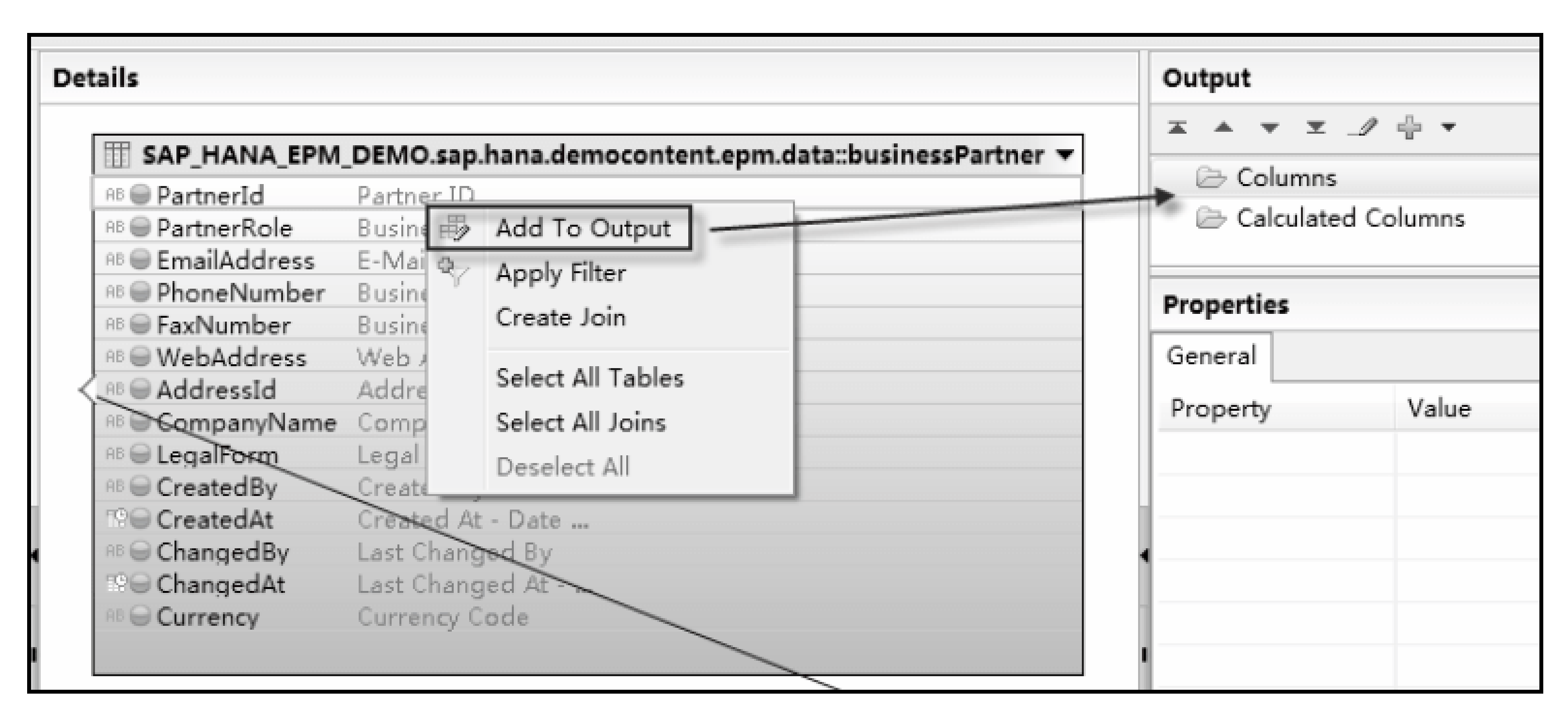


图 8-52

(9) 配置输出字段的参数。如果我们需要配置输出字段，则需要在“Output”面板中选中该字段，在底下的“Properties”面板中为其各项参数做配置，例如更改显示标签(见图 8-53)。

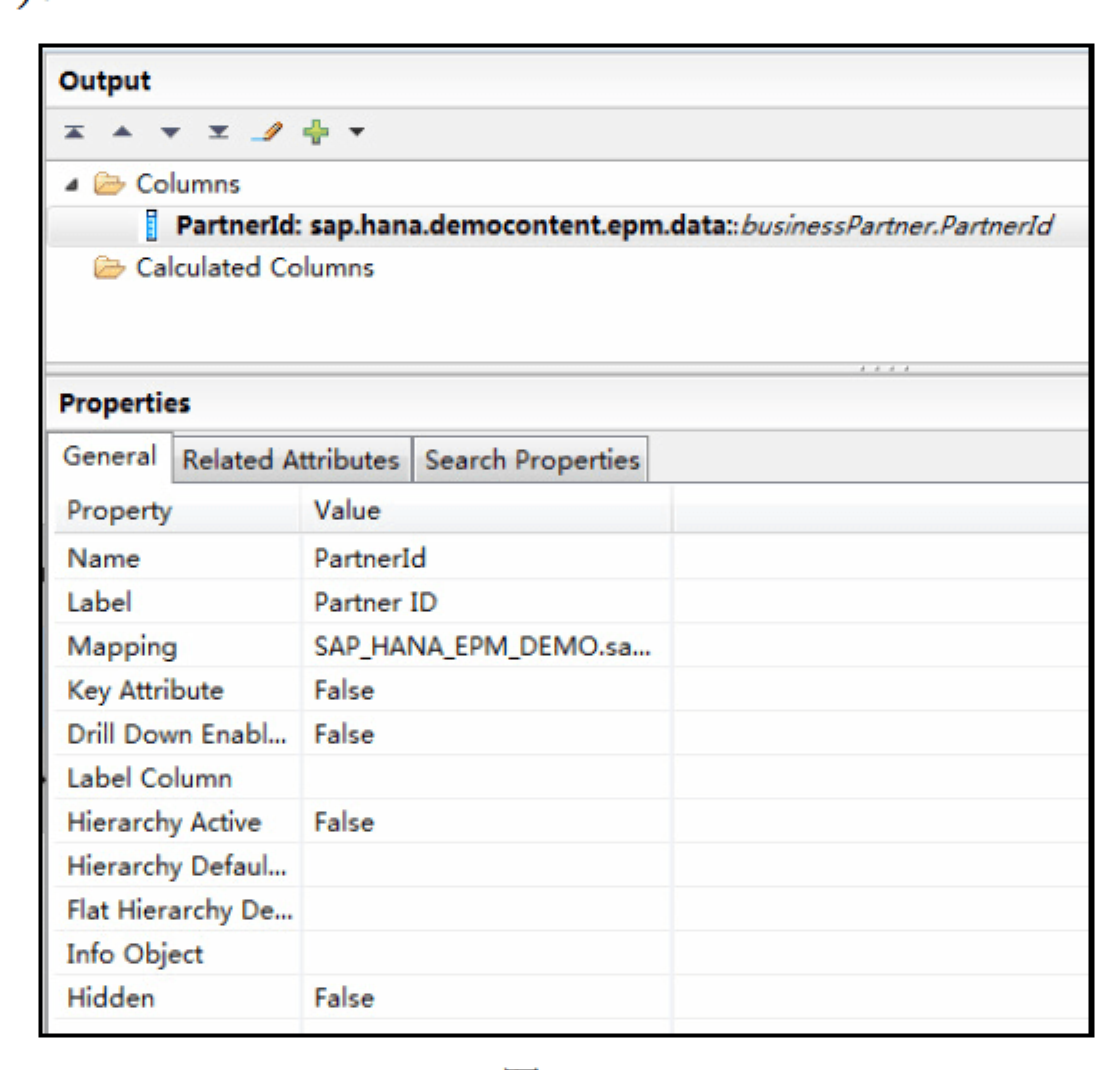


图 8-53



(10) 按照同样步骤，我们将所有需要的字段都添加到“Output”面板中，结果如图 8-54 所示。

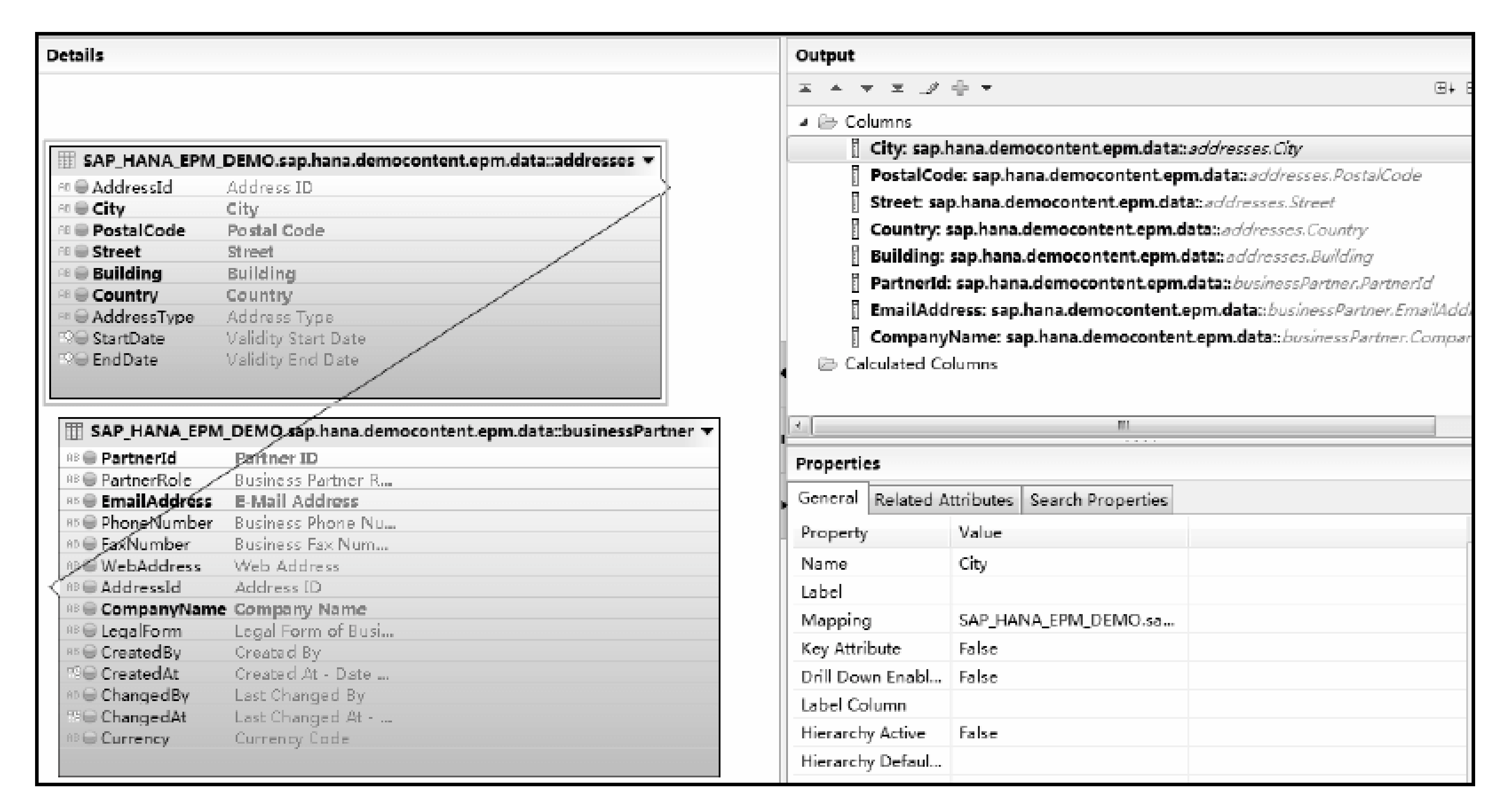


图 8-54

(11) 预览结果。单击“Data Preview”按钮预览视图输出结果(见图 8-55)。



图 8-55

(12) 在“Raw Data”标签页，你能预览视图数据，其返回值的多少由你设定的最大返回条目数决定(见图 8-56)。

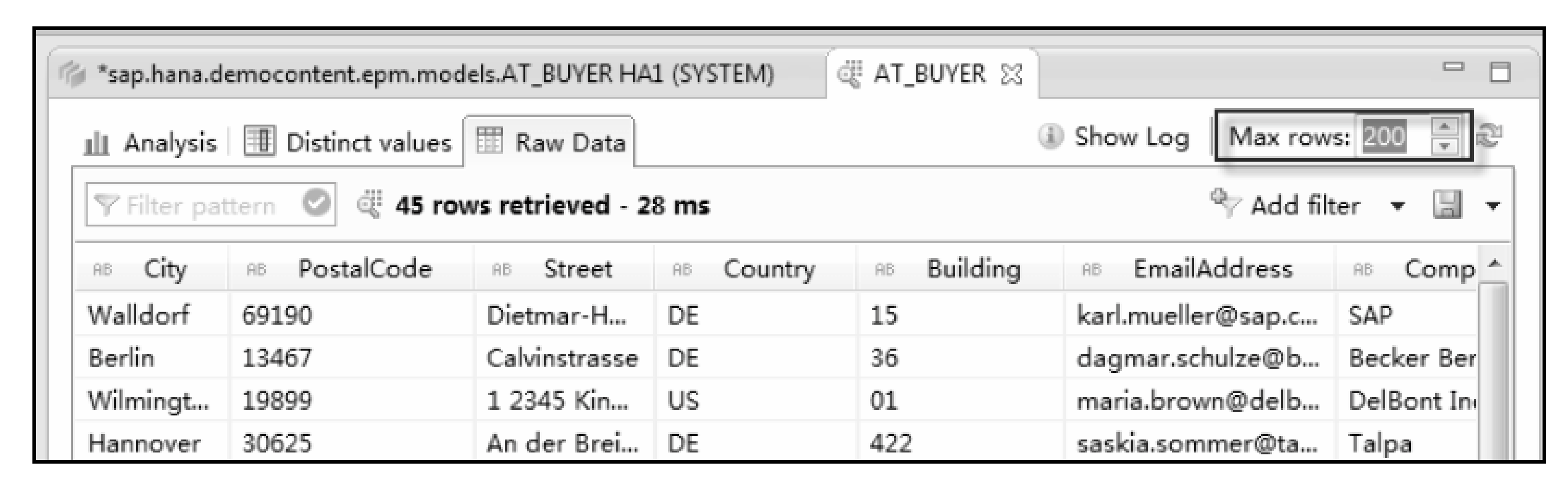


图 8-56

(13) 定义过滤器。通常来讲，我们在做分析时不需要将所有的条目都列出来，过滤器是我们常用的作为限制条件的工具。通过设置过滤器，我们可以只对我们需要的数据表条目进行分析。同样的，我们在属性视图中也可以为需要的字段定义过滤器，例如我们只对某些区域的买家进行分析，我们就需要在数据表

“sap.hana.democontent.epm.data::address”的“City”字段上设置过滤器，具体操作为：右击“City”字段调出右键菜单，选择“Apply Filter”一项。在弹出的“Apply Filter”窗口中选择操作逻辑，例如“List of values”，然后在“Value Help Dialog”窗口中选择需要分析的区域，你可以通过按住“Ctrl”键进行条目的多选操作(见图 8-57)。

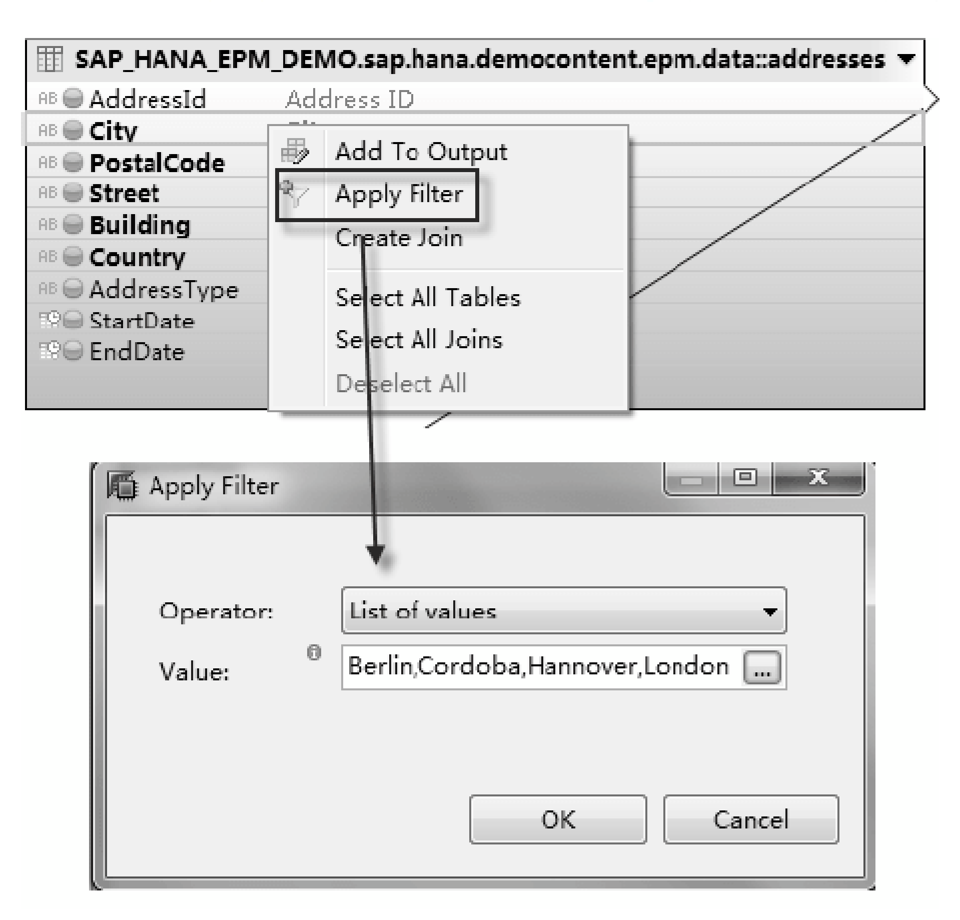


图 8-57

(14) 单击“Save and Activate”按钮保存并激活属性视图(见图 8-58)。

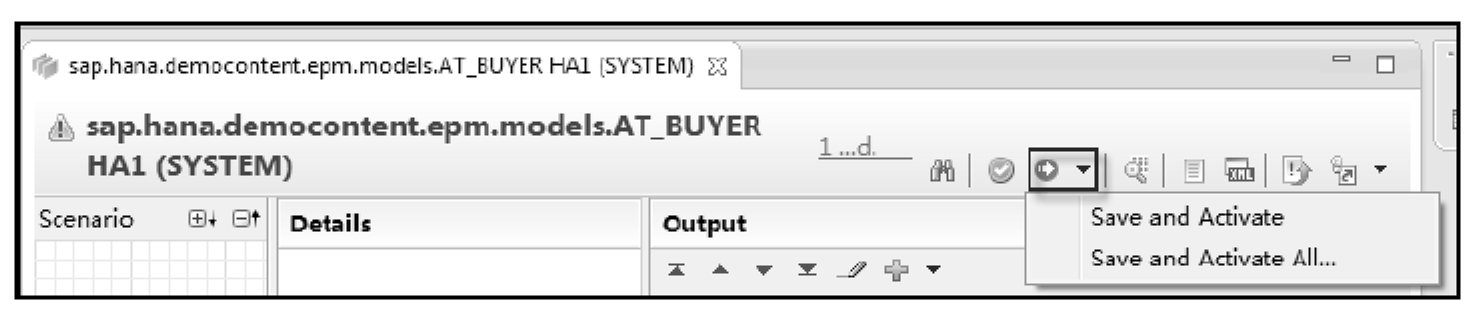


图 8-58

至此，我们就成功地创建了一个属性视图，下面我们接着来创建一个分析视图。

第五节　创建分析视图

典型的分析视图是基于包含交易数据(即度量数据)的事实数据表(Fact Table)来创建的。如果你对数据仓库有了解的话，你对事实数据表这个概念应该不陌生。为

了更好地继续分析视图的创建，我们来简单聊一下事实数据表这个概念。

一、事实数据表

通常来说，每个数据仓库都包含一个或者多个事实数据表。事实数据表可能包含企业的业务数据，如销售部门所产生的数据。事实数据表通常包含大量的行。事实数据表的主要特点是包含数字数据(Fact)，并且这些数字信息可以汇总，以提供作为分析的历史的数据。

包含在事实数据表中的“度量值”有两种：一种是可以累计的度量值，另一种是非累计的度量值。最有用的度量值是可累计的度量值，其累计起来的数字是非常有意义的。用户可以通过累计度量值获得汇总信息，例如，可以汇总具体时间段内一组商店的特定商品的销售情况。非累计的度量值也可以用于事实数据表，单汇总结果一般是没有意义的。例如，在一座大厦的不同位置测量温度时，将大厦中所有不同位置的温度累加是没有意义的，但是求平均值是有意义的。

每个事实数据表包含一个由多个部分组成的索引，该索引包含作为外键的相关性维度表的主键，而维度表包含事实记录的特性。事实数据表不应该包含描述性的信息，也不应该包含除数字度量字段及使事实与维度表中对应项的相关索引字段之外的任何数据。一般来说，一个事实数据表都要和一个或多个维度表相关联，用户在利用事实数据表创建多维数据集时，可以使用一个或多个维度表。

举个实际的例子。企业销售管理中，A 表中存放实际销售相关数据，包括客户代码、产品型号、销售金额等，B 表存放客户代码和客户名称的对应关系，C 表存放产品型号和产品名称的对应关系。则 A 是事实表，B 和 C 是维度表。

二、创建分析视图

(1) 打开 SAP HANA Studio，切换到“Modeler”透视图(见图 8-59)。



图 8-59

(2) 在“Quick Launch”视图中单击“Analytic View”一项，然后单击“Create...”按钮(见图 8-60)。

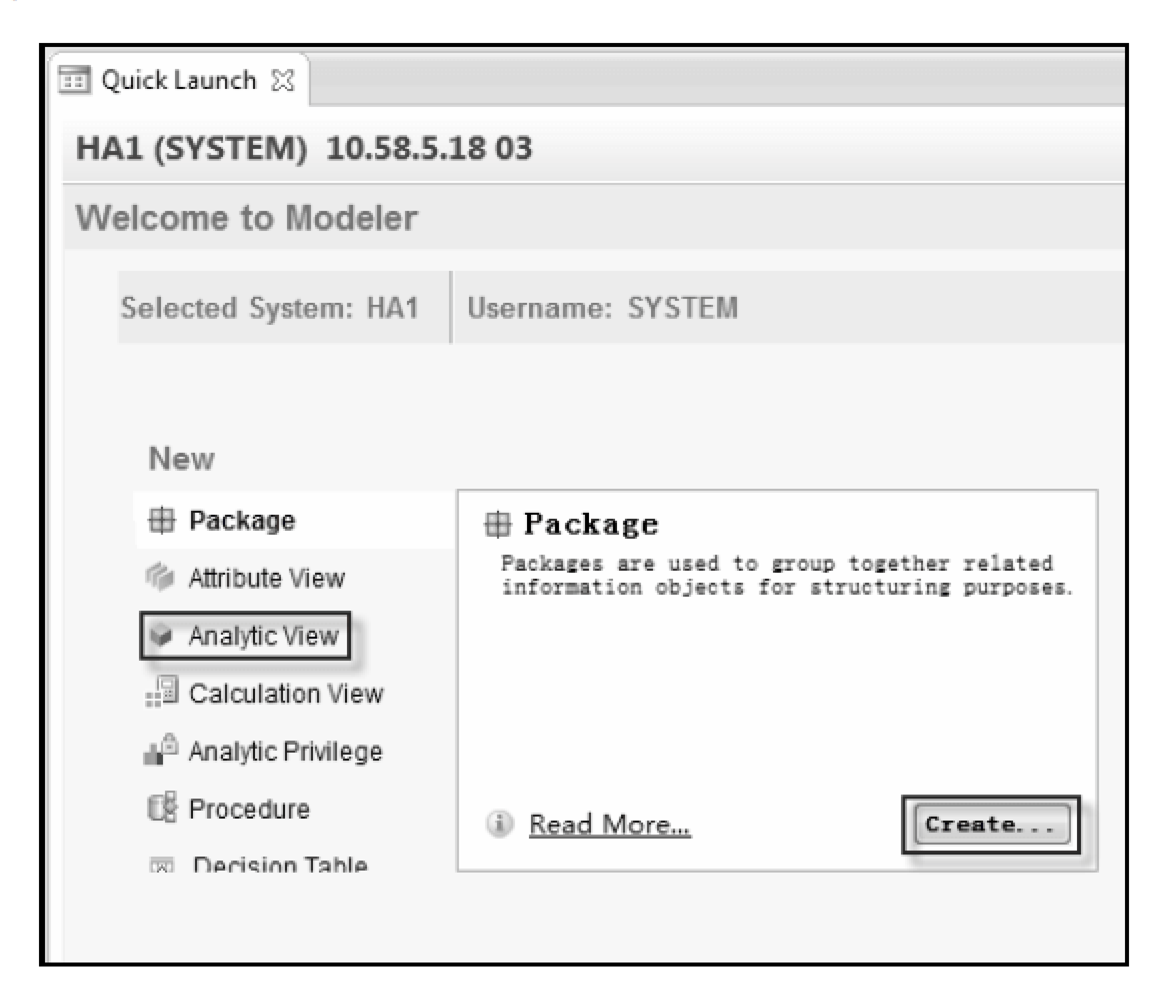


图 8-60

(3) 你还可以通过从“SAP HANA Systems”视图定位到你需要创建视图的包，右击调出右键菜单，选择“New”→“Analytic View”。本例中我们选择的包是“epm”→“models”(见图 8-61)。

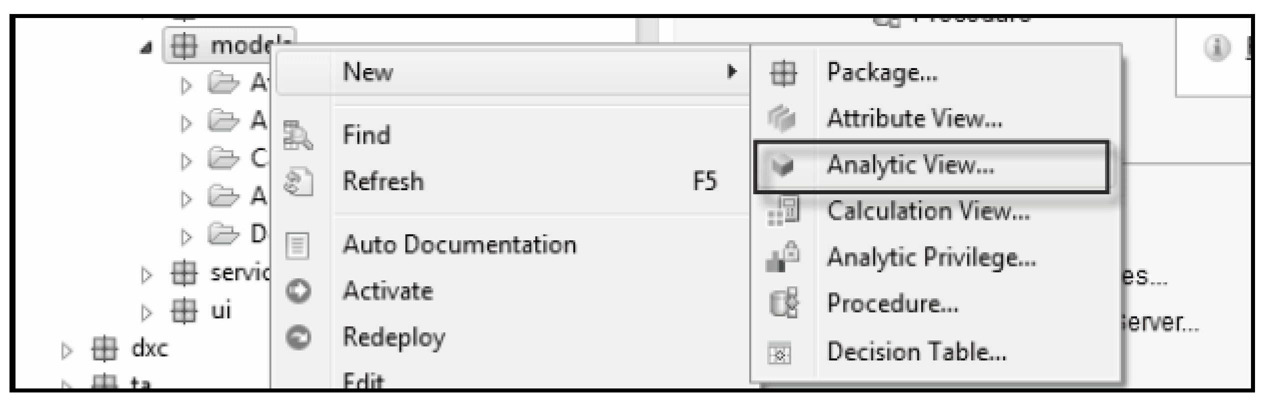


图 8-61

(4) 如下图所示在“New Information View”窗口中将分析视图的名称和描述填好。和属性视图一样，你需要为新生成的分析视图指定所属的包，如果你是基于第 3 步建立的分析视图，系统会将包路径自动带到“Package:*”字段。“View Type”字段需要选择“Analytic View”。需要注意的是，在创建分析视图时是没有“Subtype”可选的。配置好全部内容后，单击“Finish”按钮调出分析视图编辑器视图(见图 8-62)。





图 8-62

(5) 和建立属性视图一样，在分析视图编辑器编辑器中单击“Data Foundation”节点，将需要的数据表加入“Details”面板中(见图 8-63)。

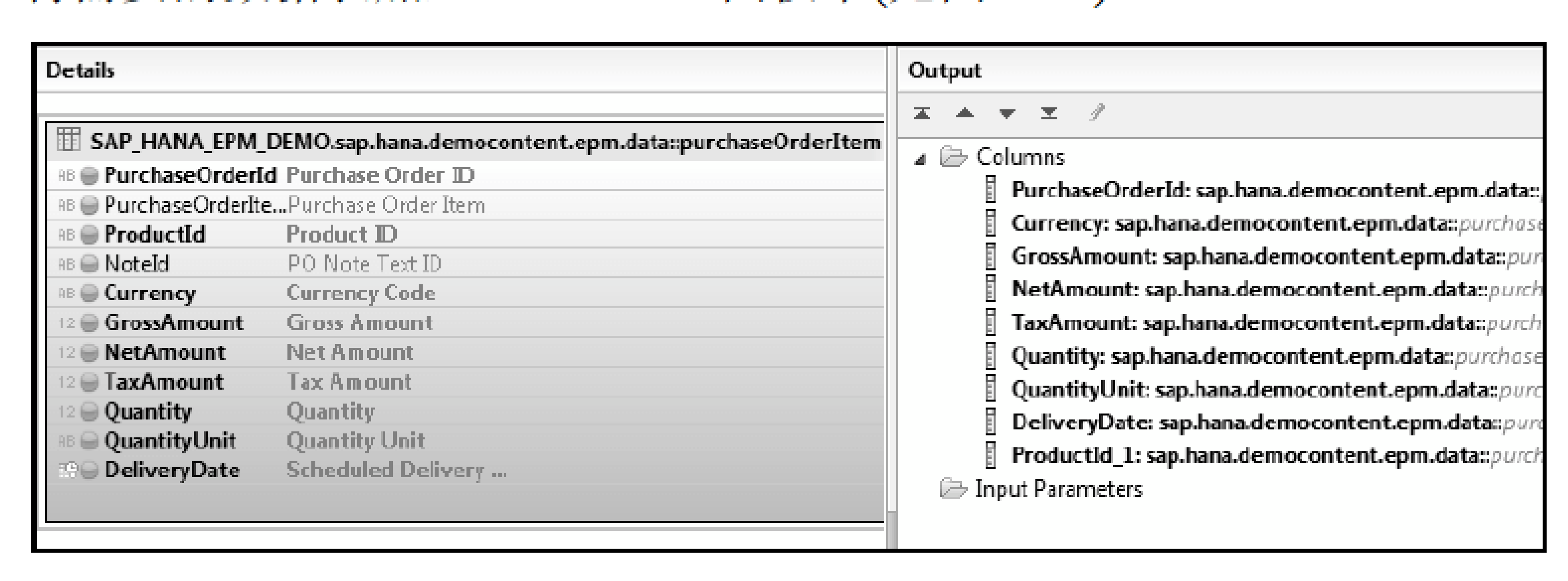


图 8-63

在本例中，我们使用 EPM 案例中的采购相关的事实表“PurchaseOrderItem”来建立分析视图。将该表拖入“Details”面板后，请参照和属性视图一样的操作将上图中标出的字段添加到“Output”面板。由此我们可以看出分析视图不同于属性视图需要多个表连接才能建立，它可以基于一个单独的事实表来建立，也可以通过不同的表(事实表与维度表)做连接来建立。

与属性视图一样的，你可以在事实表字段上面通过右击“Apply Filter”建立过滤器来限制分析视图的输出内容。

(6) 单击“Logical Join”节点，将 EPM 自带的属性视图“AT_PRODUCT”拖入

“Details”面板(见图 8-64)。

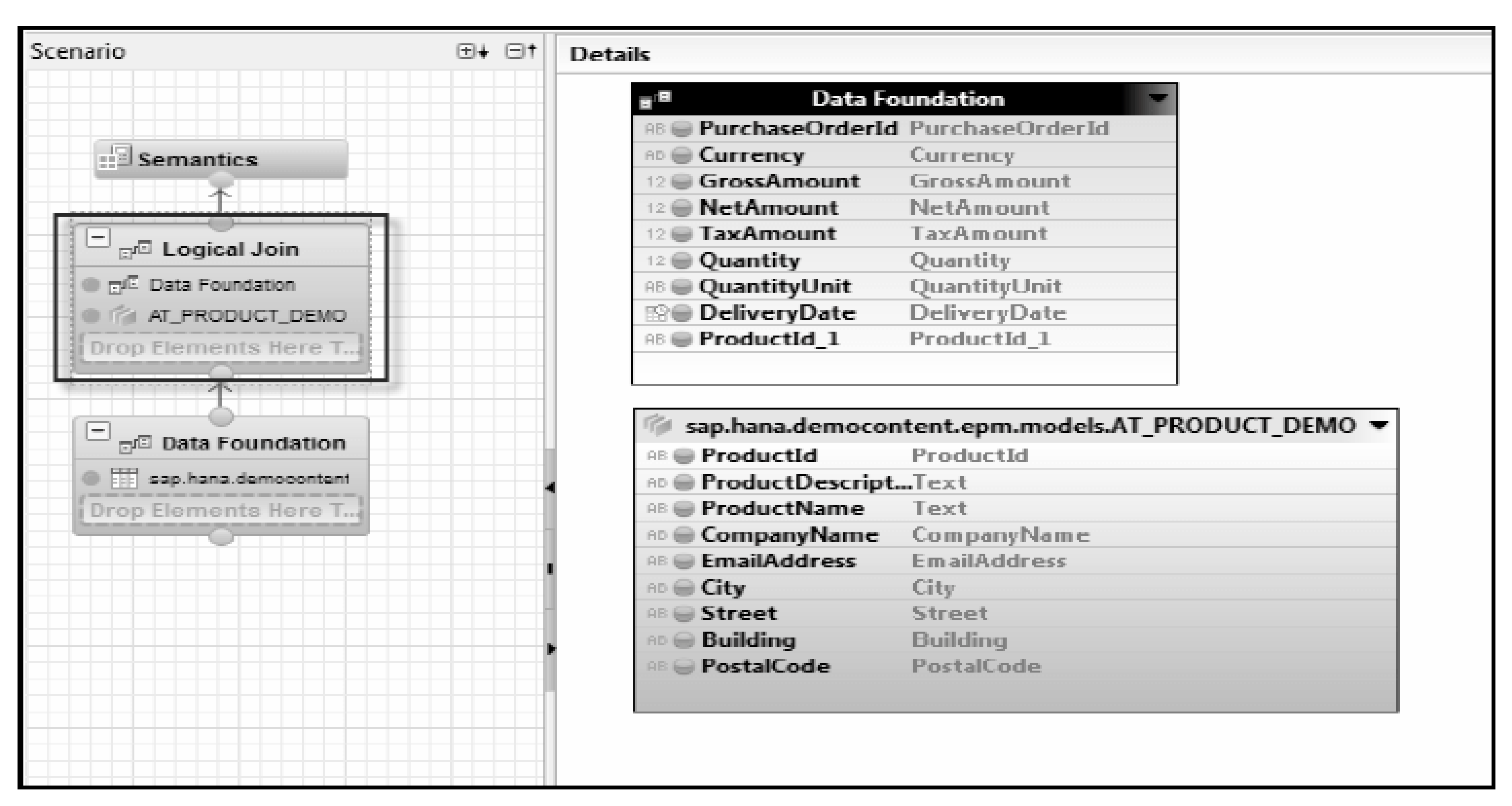


图 8-64

在此我们介绍一下这个新出现的节点“Logical Join”。这个节点是用来定义在“Data Foundation”节点中选中的事实表输出字段和属性视图之间关系的地方，即我们可以在这里创建星形模式(Star Schema)。这说明分析视图不仅可以使用数据表，还可以使用属性视图来作为它组成的一部分。

(7) 在“Data Foundation”区块下右击“ProductId_1”字段，选择“Create Join”，调出“Create Join”窗口，按照图 8-65 所示填写各字段内容。



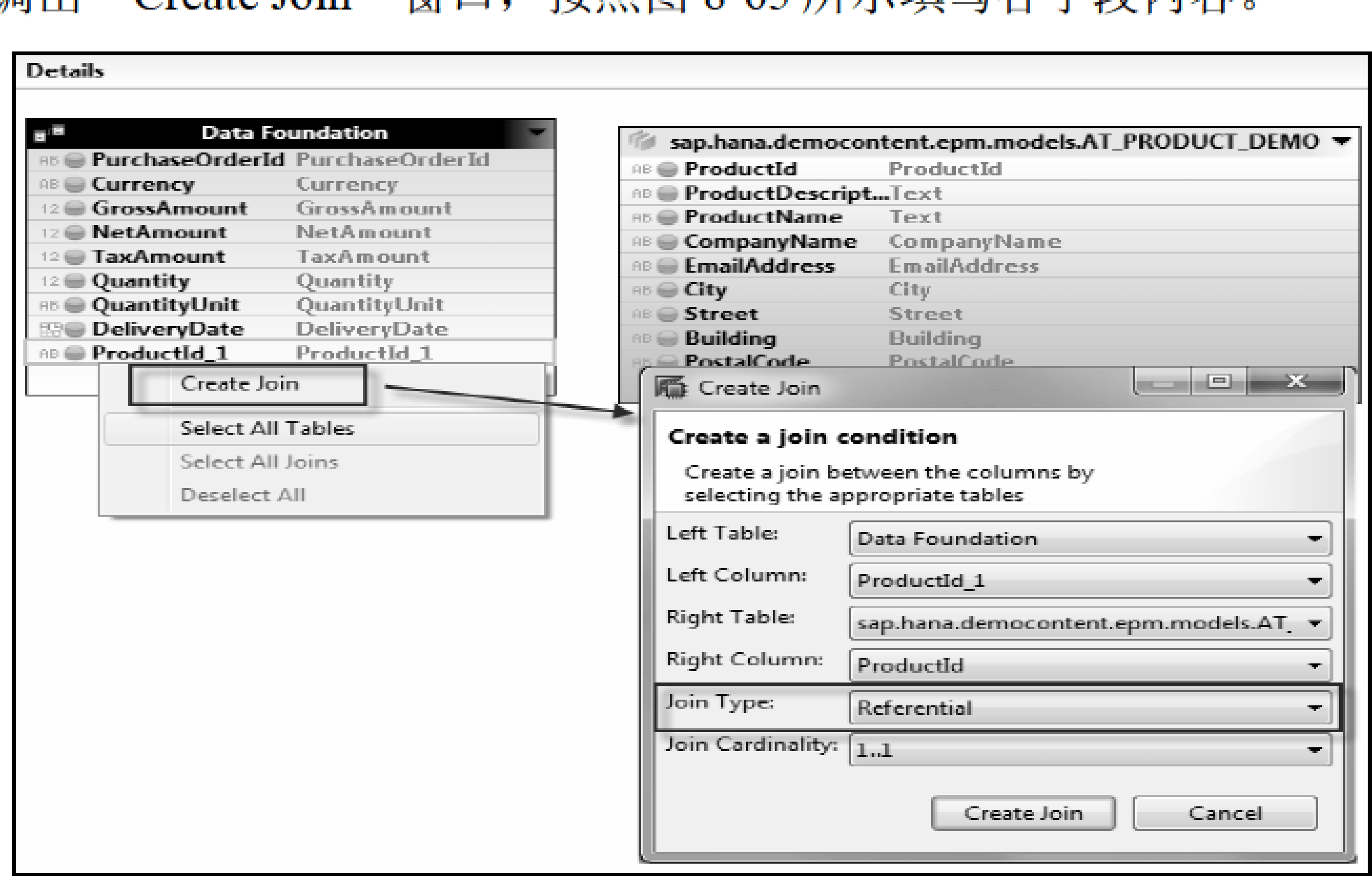


图 8-65

在“Create Join”窗口中我们唯一陌生的值就是在“Join Type”中出现的“Referential”连接类型。我们在本章开头介绍连接类型时曾提到它，现在我们就对此连接类型做进一步说明。

从语法上来讲，“Referential”连接类型其实就是一种内连接(Inner Join)，但是这种内连接是在假设左表中的数据都能在右表找到的基础上实现的，比如我们在本步骤中的定义就是假设左表所包含的所有“Product_Id”都能在右表中找到。因此当我们使用“Referential”连接类型时，系统不会检查右表中的数据而会直接执行内连接操作，这样就可以大大提高表之间内连接查询的速度。在我们能够保证左表中用于连接的字段所包含的数据都能在右表中找到时，“Referential”是我们推荐的连接类型。

当我们不能够保证左表中用于连接的字段所包含的数据始终都能在右表找到时，我们就应该慎重使用“Referential”连接类型，因为如果在这种情况下使用“Referential”连接类型可能会造成系统错误。通常来讲，在这种情况下我们应考虑使用内连接或者左外连接类型来代替。

(8) 单击“Semantics”节点。在“Column”面板中检查/修改度量属性字段的聚合方式(见图 8-66)。

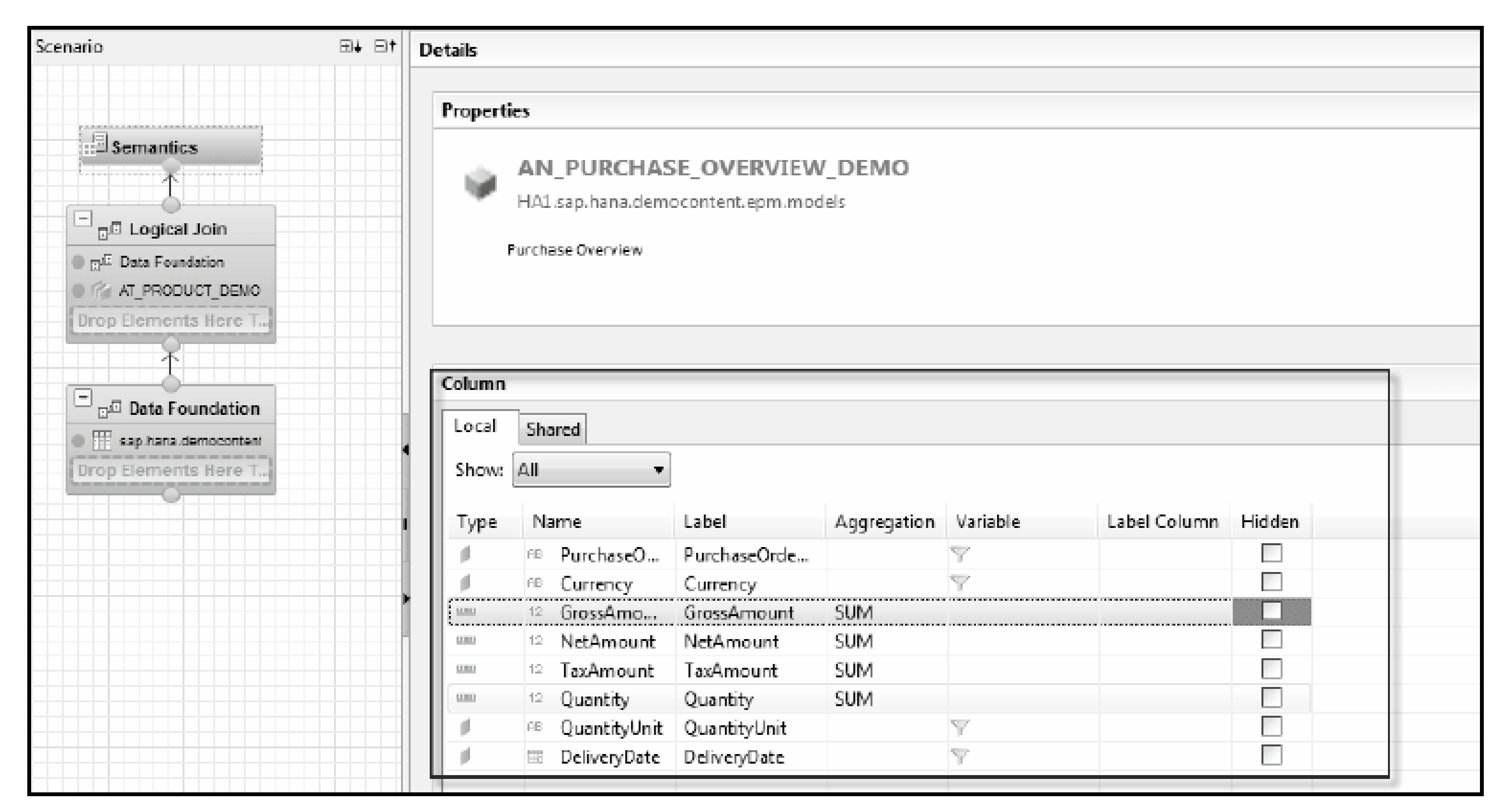


图 8-66

由上图可以看出，对于度量属性的字段，如“GrossAmount”、“NetAmount”等，分析视图会自动赋予其“SUM”的聚合方式(Aggregation)。如果你需要使用其他的聚合方式，只需要单击度量属性字段所对应的“Aggregation”字段，在下拉列

表中选择一种合适的聚合方式即可(见图 8-67)。

12	GrossAmo...	GrossAmount	SUM	
12	NetAmount	NetAmount	SUM	
12	TaxAmount	TaxAmount	MAX	
12	Quantity	Quantity	MIN	
AB	QuantityUnit	QuantityUnit	COUNT	

图 8-67

(9) 在“Variables/Input Parameters”面板为属性字段定义变量。为属性字段定义变量是为了在运行分析视图时对属性字段进行过滤，即在输出数据中只出现符合我们需要的属性数值。本例中我们对货币单位定义变量(如图 8-68 所示)以使输出条目只出现由美元结算的采购订单，具体操作如下：

在“Variables/Input Parameters”面板单击“+”按钮，选择“Create Variable”，在调出的“New Variable”窗口中输入变量的名称(Name:*)和标签(Lable)。在“Attribute:*”字段中选择“Currency”，在“Selection Type”下拉列表中选择“Single Value”，“Default Value”为“USD”，并在“Apply the variable filter to”区域通过“Add”按钮将此变量赋予“Currency”属性字段。

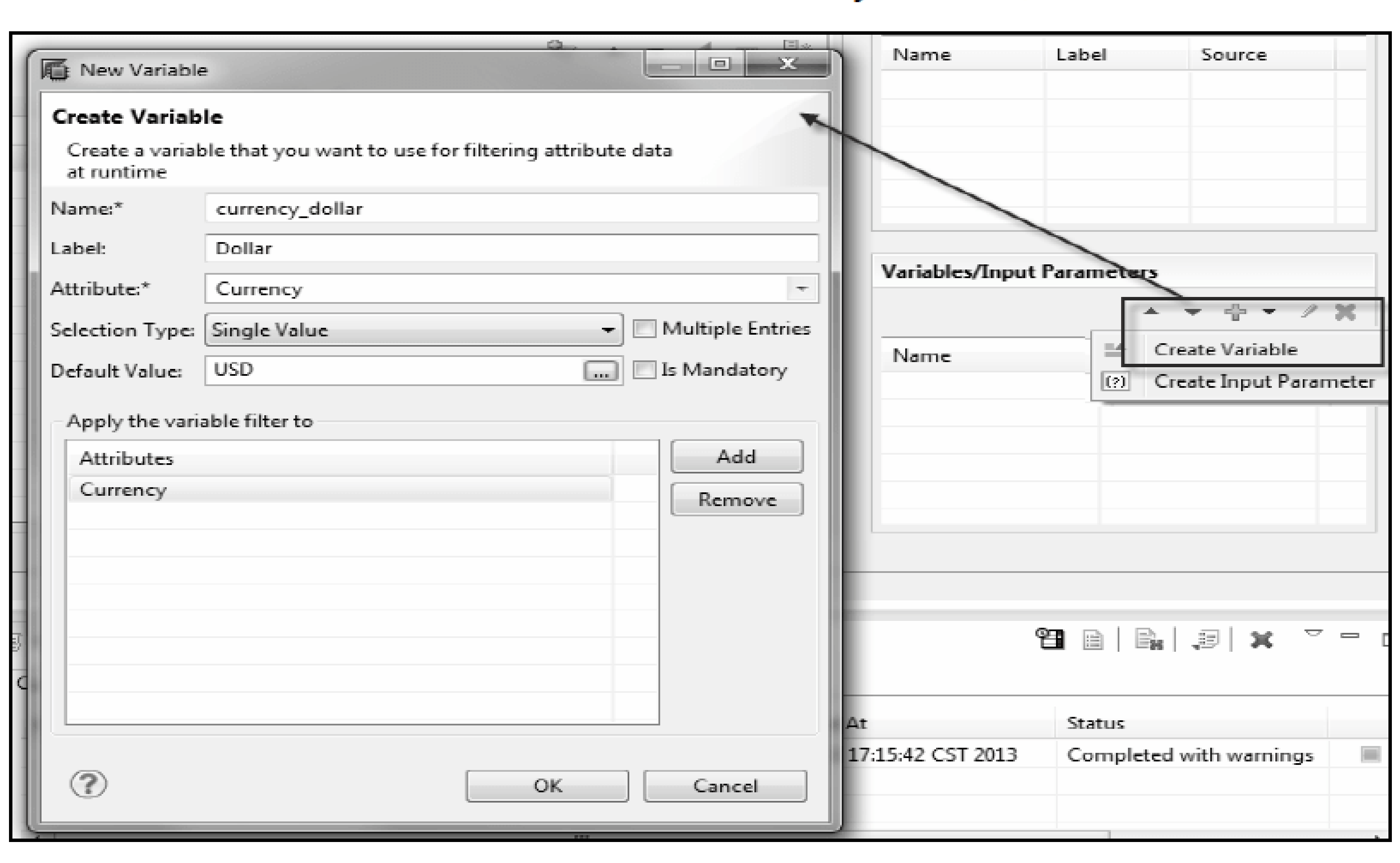


图 8-68



返回“Column”面板你就能看见“Currency”字段已经包含变量“currency_dollar”(见图 8-69)。单击“Save and Activate”按钮保存并激活分析视图(具体操作同属性视图)。

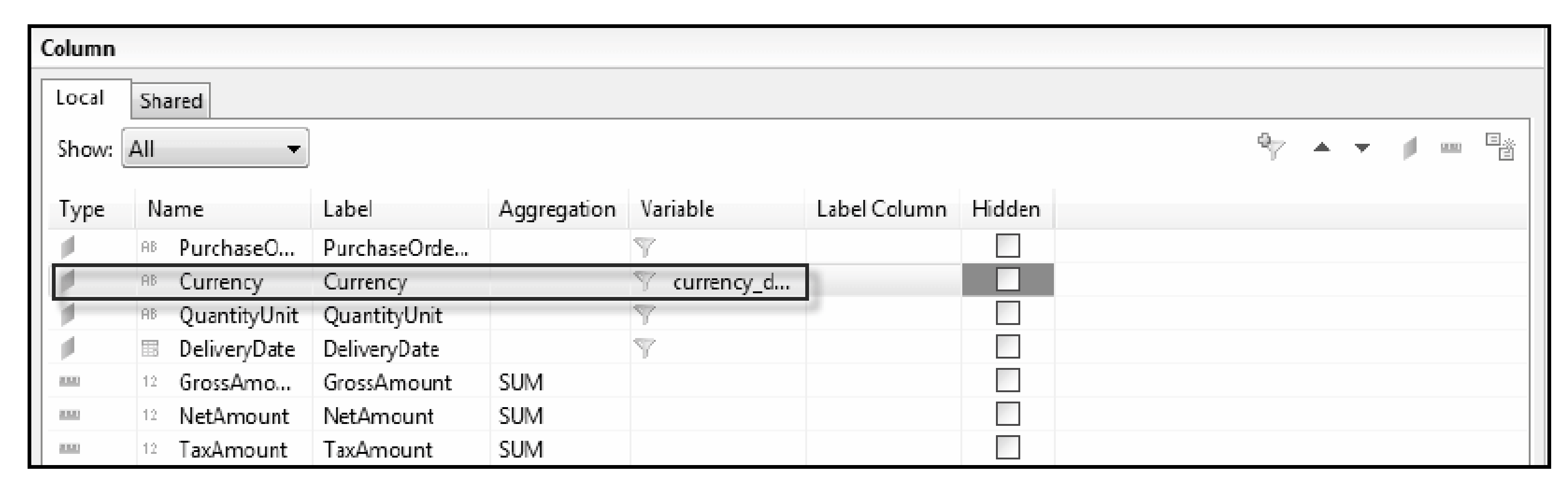


图 8-69

如果你在添加变量前和添加变量后使用数据预览功能检查过新建的分析视图的输出数据，你就能得到如下比较结果：

- “Currency”字段没加变量的输出结果如图 8-70 所示。

ProductDescription	PurchaseOrderId	Currency	QuantityUnit
High resolution 320x320 t...	0300000765	MXN	EA
Complete package. 1 Use...	0300000799	EUR	EA
7 LCD Screen. storage bat...	0300000293	EUR	EA
17 Optimum Resolution ...	0300000618	EUR	EA
Complete package. 1 Use...	0300000760	USD	EA
Print up to 25 ppm letter ...	0300000581	EUR	EA

图 8-70

- “Currency”字段添加变量的输出结果如图 8-71 所示。

ProductDescription	PurchaseOrderId	Currency	QuantityUnit
Connecting our beamers ...	0300000042	USD	EA
Complete package. 1 Use...	0300000018	USD	EA
Our new multifunctional ...	0300000443	USD	EA
Connecting our beamers ...	0300000046	USD	EA
23 Optimum Resolution ...	0300000065	USD	EA
Connecting our beamers ...	0300000296	USD	EA
Complete package. 1 Use...	0300000511	USD	EA
Hurricane GX: DDR2 RoH...	0300000714	USD	EA
Our new multifunctional ...	0300000693	USD	EA

图 8-71

在创建分析视图时，我们可以为属性字段生成不同的变量来使输出数据最小化，提高视图的执行效率。属性变量不仅可以是单独的数值，也可以是某一间隔(例如 2000—2005 年度的采购订单)或者范围内(例如采购金额≥1000 美元)的数值。在运行阶段，我们可以通过将不同的变量赋予属性字段来动态显示不同的分析数据。

(10) 为引用的属性视图定义输出字段。在“Column”面板中切换到“Shared”标签页即可看到我们引用的属性视图的全部字段。你可以在这里为这些字段设定是否

为隐藏。同时也可以为这些字段如第 9 步一样定义变量。这里我们将“ProductId”字段隐藏(见图 8-72)。

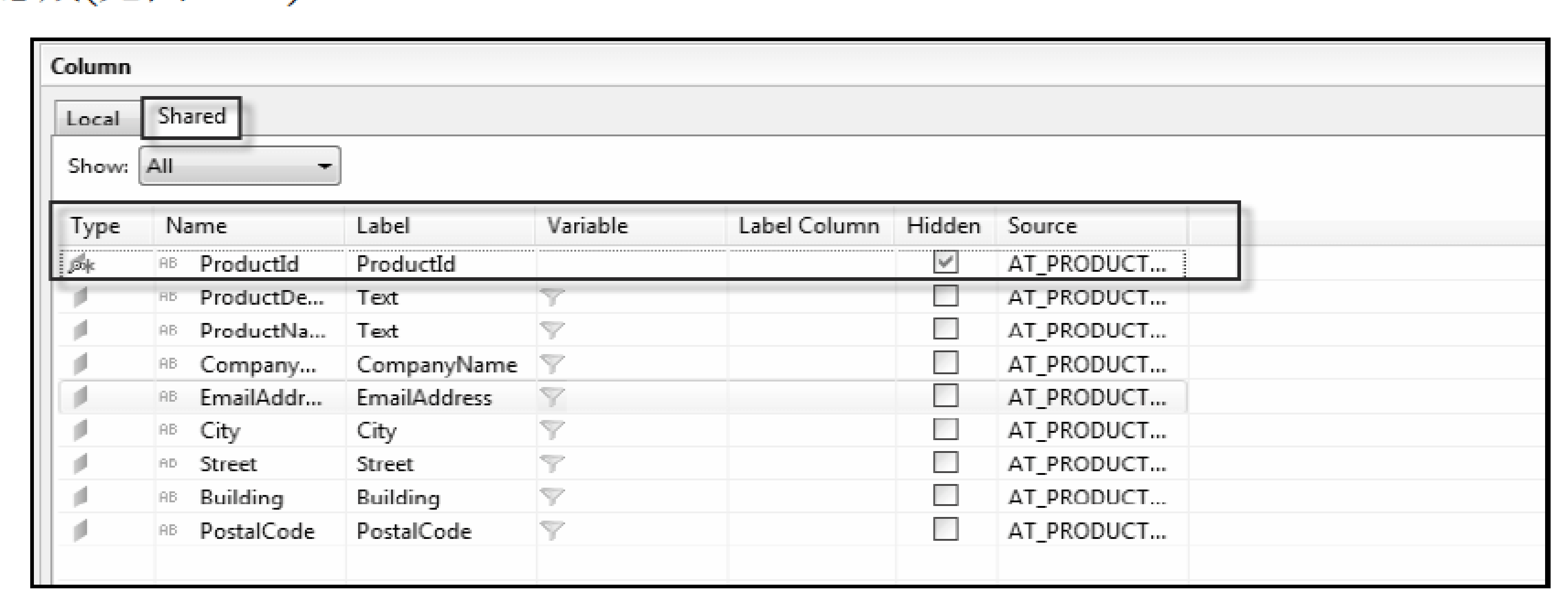


图 8-72

(11) 在数据预览视图检查结果。保存并激活分析视图后，我们就能通过单击“Data Preview”按钮预览数据(见图 8-73)。

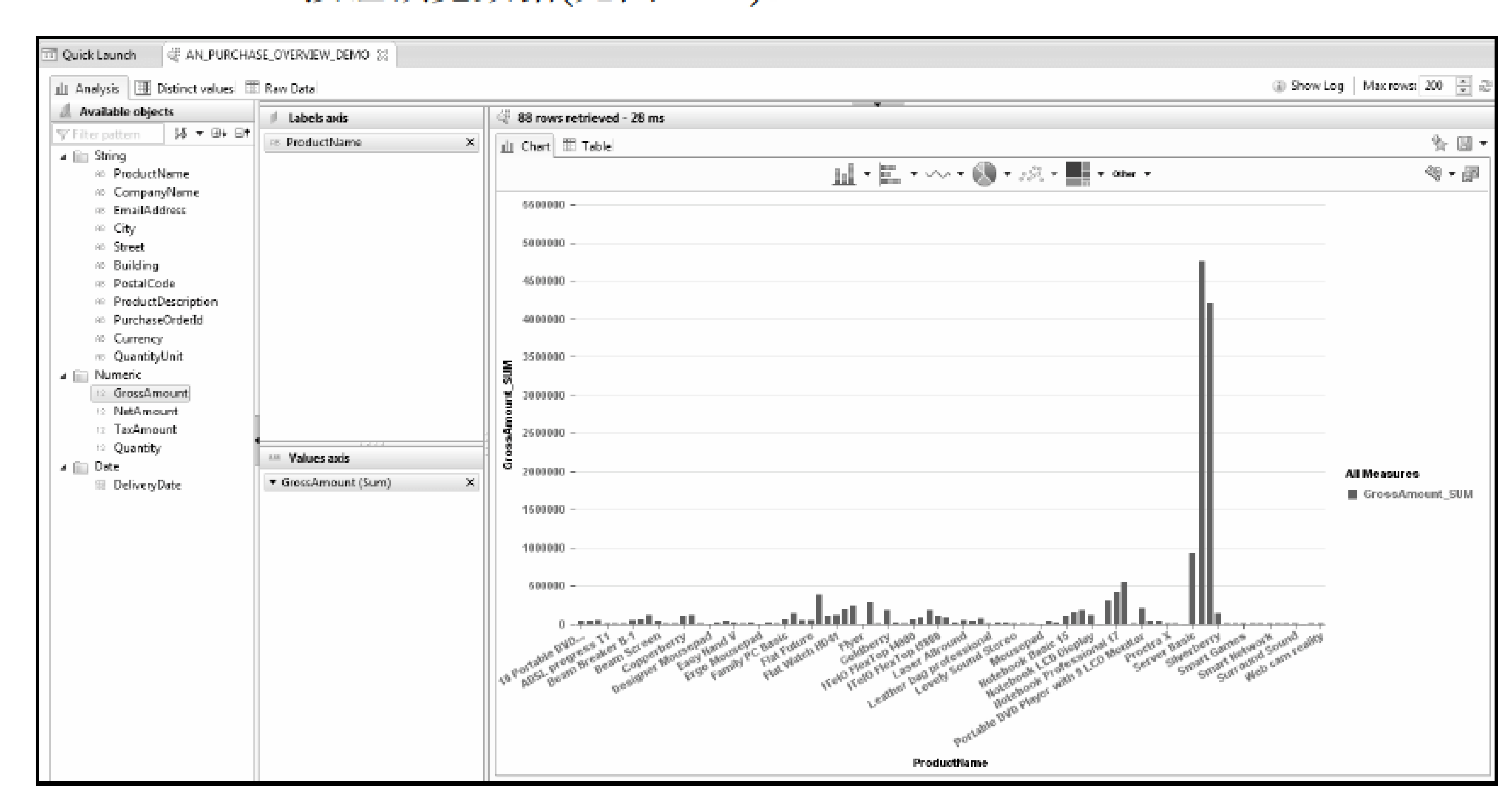


图 8-73

SAP HANA Studio 提供了多种图表来显示分析视图的结果，如图 8-73 所示我们使用柱状图来显示采购产品和采购产品总金额的关系，我们还可以通过切换到饼状图来显示具体百分比(见图 8-74)。



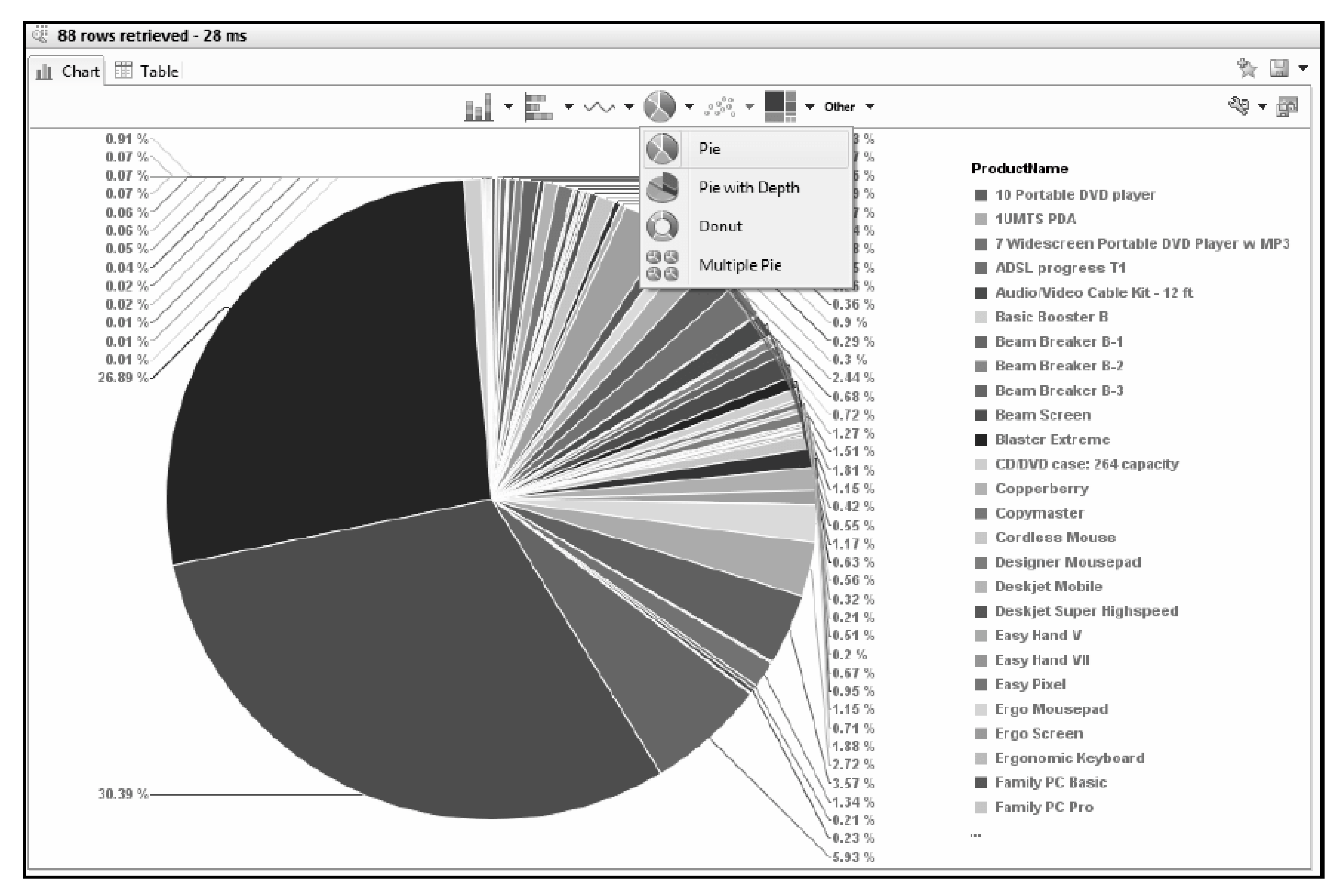


图 8-74

至此，在 SAP HANA Studio 中创建分析视图的基本操作已经完成，接下来我们看看计算视图的创建方法。

第六节　创建计算视图

一、创建计算视图的准备

前面介绍的 SAP HANA 的属性视图和分析视图为我们提供了很好的分析数据服务支持。通常来说如果我们需要取得的分析数据逻辑不是很复杂，基本上这两种视图的组合都能满足我们的需要。但是当我们需要取得的分析数据包含非常复杂的逻辑运算时，我们就应该考虑另一种视图，即计算视图。计算视图在 SAP HANA 中是为了定义更高级的 SAP HANA 数据库数据切片而使用的。计算视图不仅能够提供属性视图和分析视图的全部功能，还能提供更加复杂的逻辑运算功能。

计算视图可以包含度量数据并用于多维度报表，也可以不包含度量数据(仅包含

属性数据)以用于陈述性报表。你可以使用 SAP HANA Studio 自带的图形化工具来生成计算视图，也可以通过 SAP HANA Studio SQL 控制台来生成计算视图。这些多种多样的组合使得计算视图能够非常灵活地用于绝大多数复杂的业务需求。

与属性视图和分析视图一样，计算视图也可以通过“Quick Launch”视图快速创建，下面我们来看看具体的创建步骤。

(1) 启动 SAP HANA Studio，在“Quick Launch”视图中单击“Calculation View”并单击“Create”按钮(见图 8-75)。

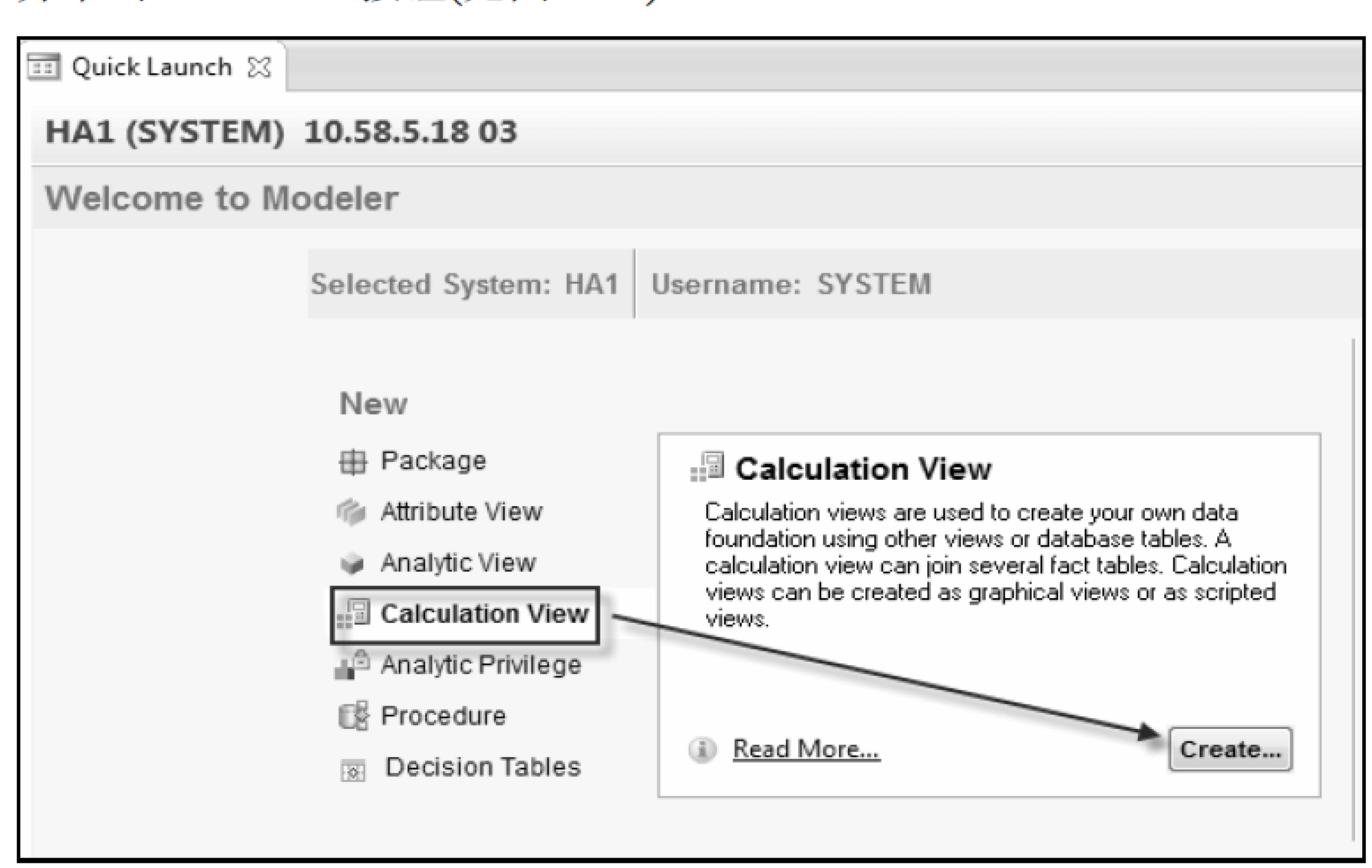


图 8-75



同样的，你也可以通过和创建属性视图及分析视图一样的方法，在“Catalog”的展开指定包层直接建立计算视图。这里我们就不再详述具体步骤了。

(2) 在调出的“New Calculation View”窗口中为新的计算视图输入名称(Name:*)和描述(Description)。输入完毕后为新计算视图指定所属包，本例中我们指定的包为“sap.hana.democontent.epm.models”。如果你把图 8-76 所示的向导窗口和新建属性视图及分析视图的向导窗口进行比较的话，你会发现 SAP HANA Studio 专门为计算视图提供了一个全新的向导窗口，在这个窗口里面你可以选择不同的方法来创建计算视图。

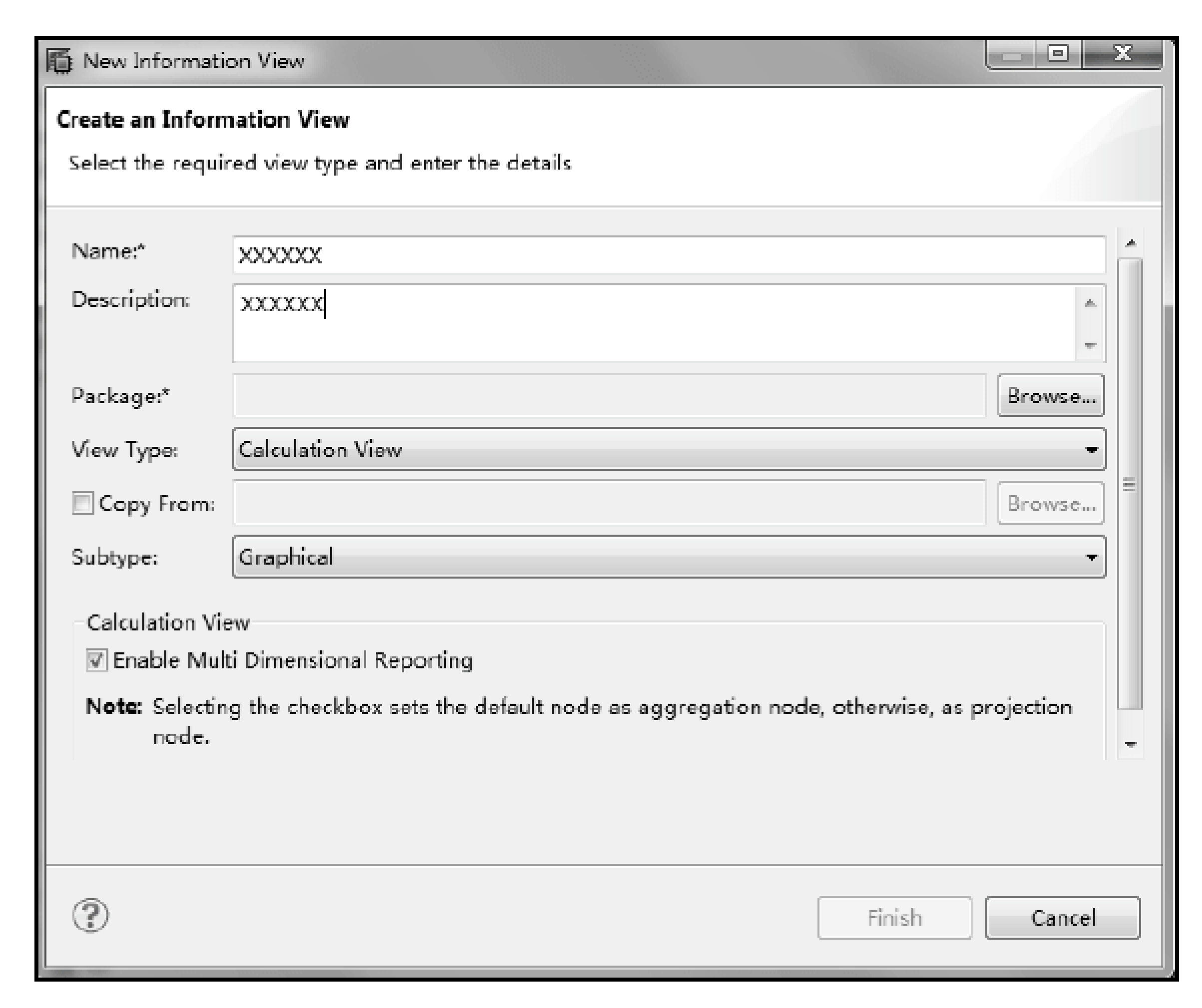


图 8-76

如果你是创建全新的计算视图，请选择“Create New”选项。如果你想基于已经存在的计算视图来创建新的视图，则需要选择“Copy From”选项并通过右边“Browse”按钮指定被复制的计算视图的路径。

“Subtype”选项包含“Graphical”(图形化)和“SQL Script”(SQL 脚本)两种类型，正如我们前面说过的，我们可以建立基于“SQL Script”的计算视图，也可以通过图形化工具建立计算视图。因此我们下面将分别介绍这两种创建方式的不同。

二、创建基于 SQL 脚本的计算视图

(1) 在“Subtype”选项里面选择“SQL Script”(见图 8-77)。



图 8-77

(2) 选择正确的包(见图 8-78)。

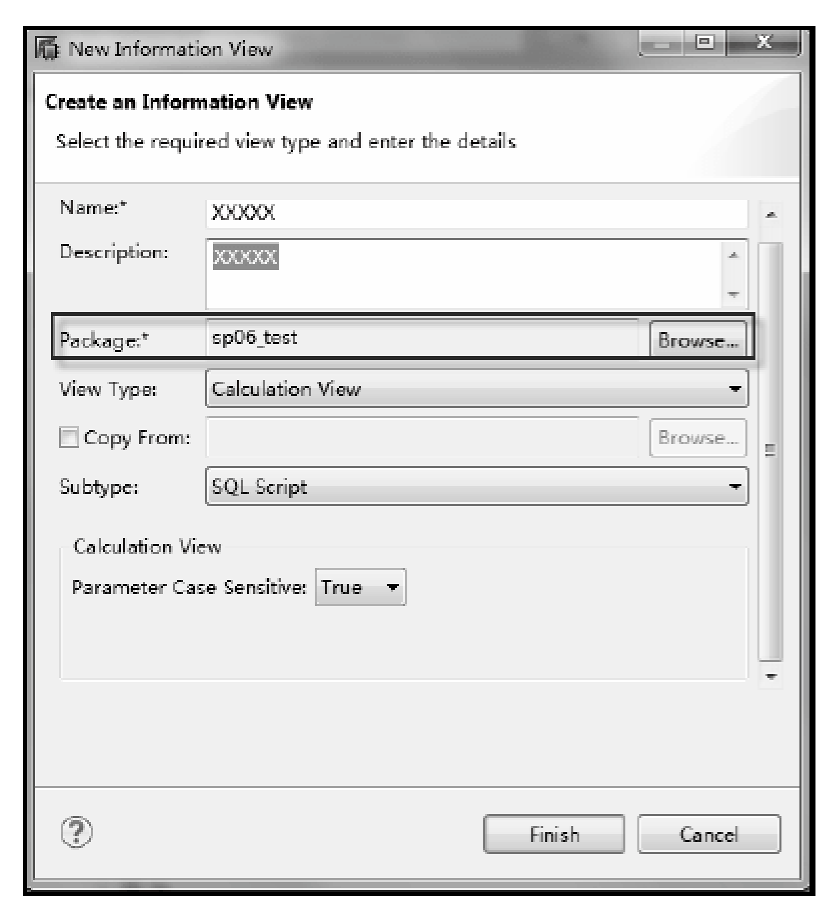


图 8-78

(3) 单击“Finish”按钮。

(4) 编写 SQL Script。如图 8-79 所示，在“Scenario”面板下单击“Script_View”节点，在右边的“Details”面板中“BEGIN”和“END”段落内输入相应的 SQL Script(我们将在第十章详细介绍 SQL Script)。

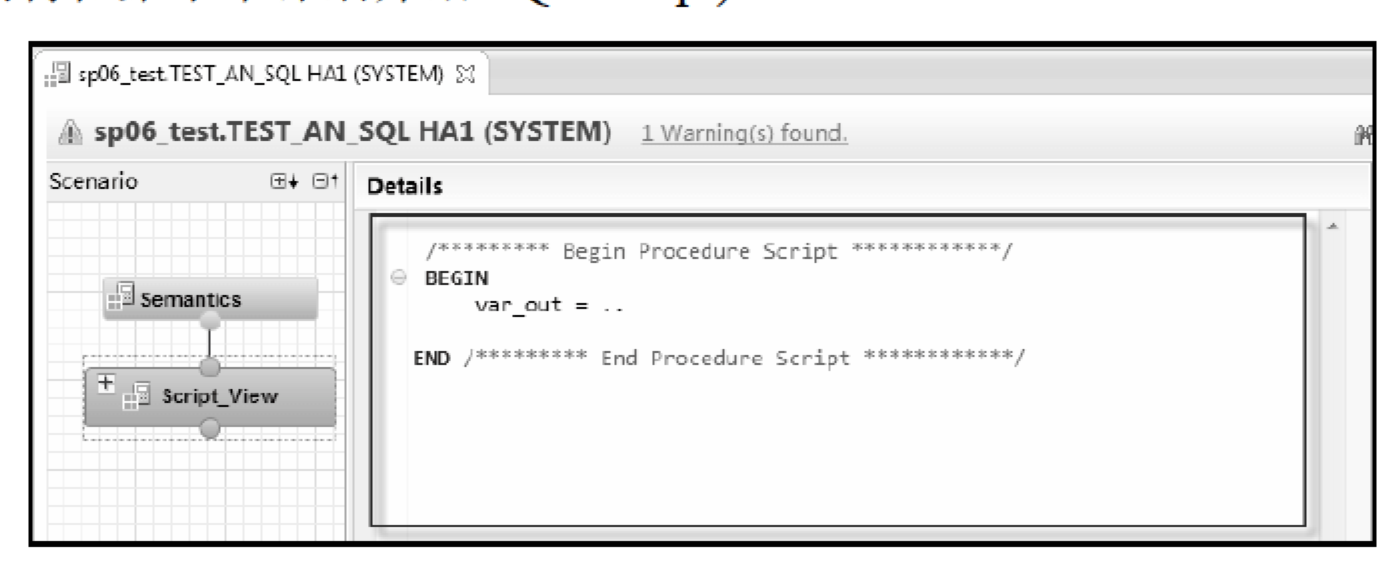


图 8-79

(5) 定义输出字段。在“Output”面板中点击“+”按钮，选择“Create Target”一项(见图 8-80)。

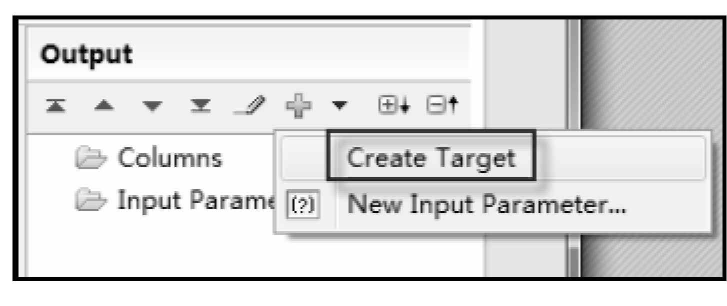


图 8-80



(6) 单击“+”按钮，依次将输出字段定义好(见图 8-81)。

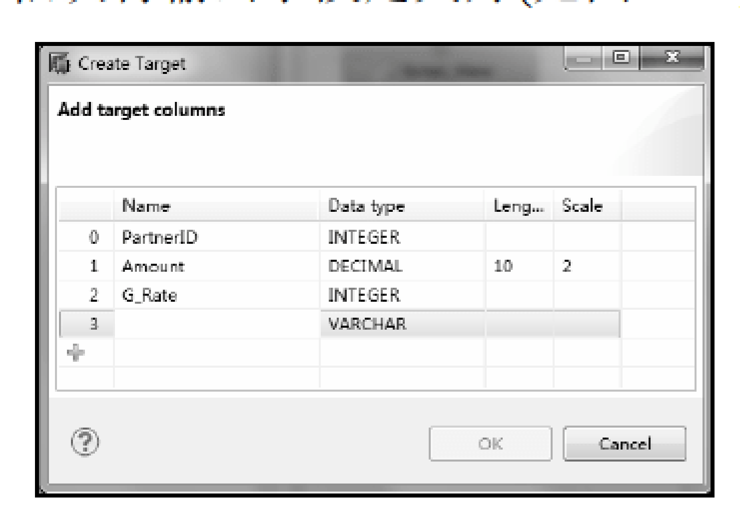


图 8-81

(7) 定义输入参数。右击“Input Parameters”节点，选择“New”(见图 8-82)。

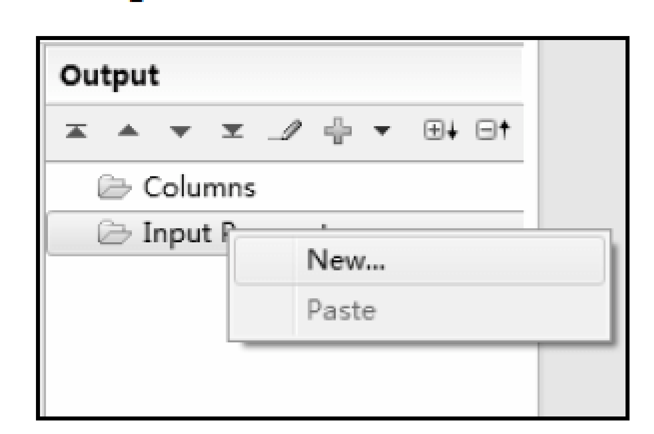


图 8-82

(8) 在图 8-83 所示的界面中为新计算视图创建一个输入参数。

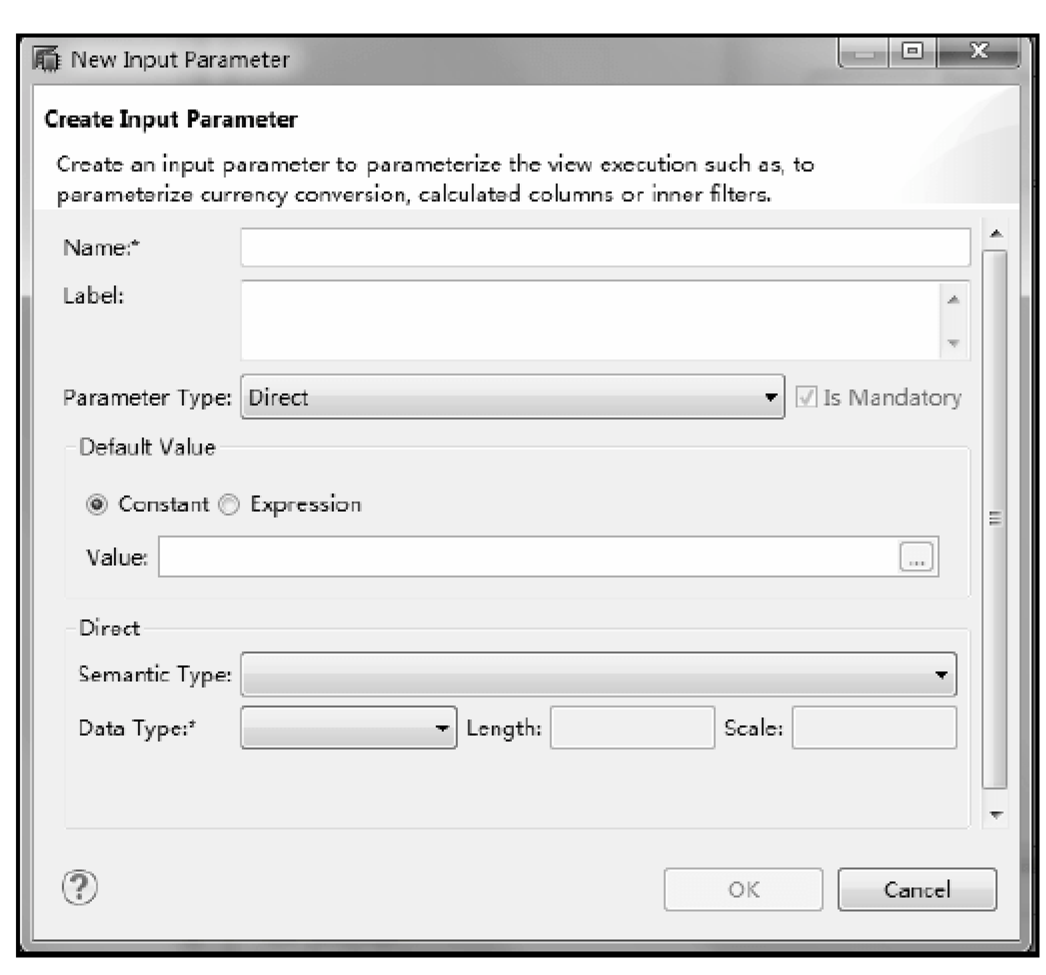


图 8-83

(9) 定义属性字段和度量字段。单击“Semantics”节点，在“Column”面板下

我们可以看到我们刚才定义的所有输出字段。在这里，我们能为每个字段定义其为属性字段或者度量字段，同时我们还能在字段上面定义层次结构(见图 8-84)。

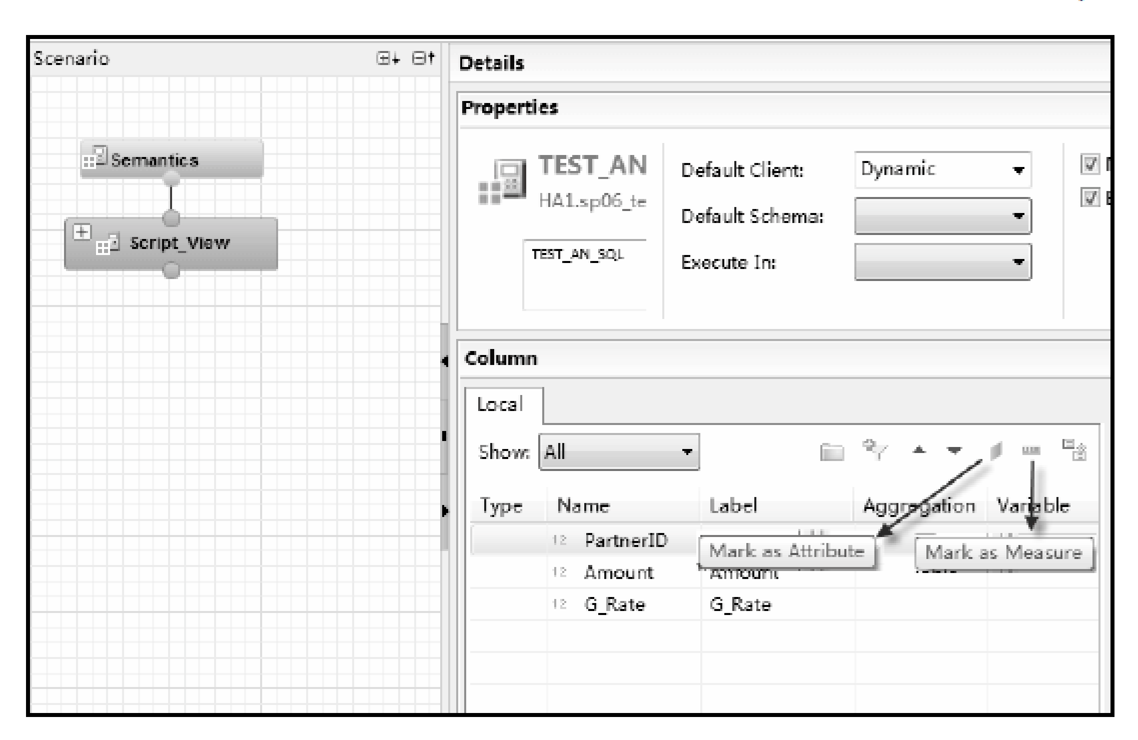


图 8-84

(10) 保存并激活新计算视图，单击“Data Preview”预览结果。

三、创建基于图形化工具的计算视图

(1) 在“Quick Launch”视图中单击“Calculation View”→“Create”调出计算视图生成向导视窗，按照图 8-85 所示为新计算视图输入名称和描述，选择所属的包。

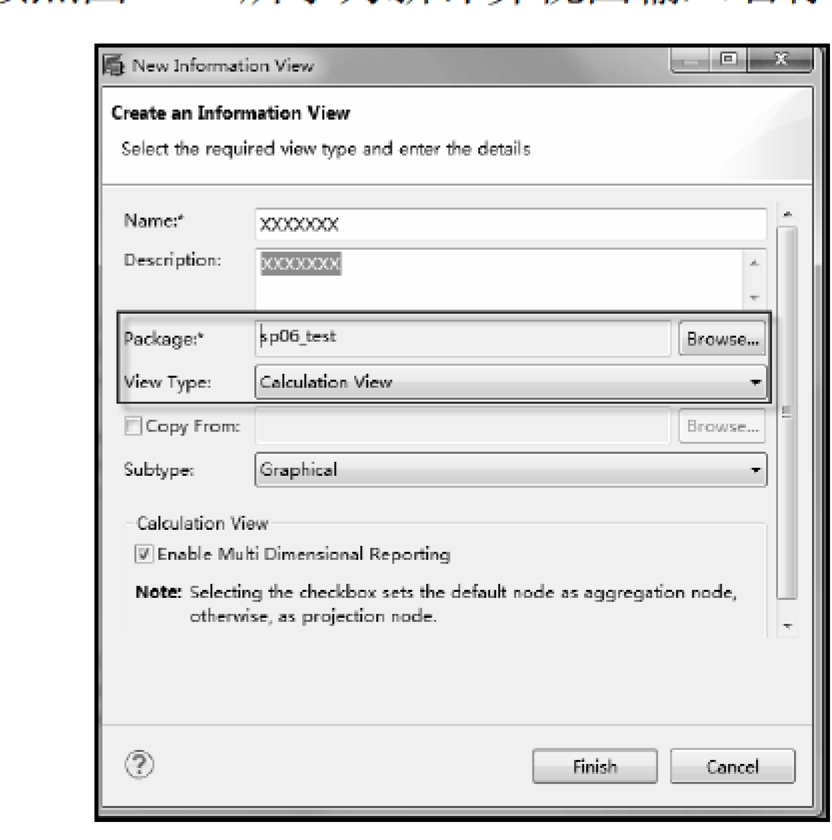


图 8-85

我们在这一步将“Subtype”选项改为“Graphical”，即图形化。最后单击“Finish”按钮。

(2) 在“Scenario”面板中，选择你需要的分析工具，目前 SAP HANA Studio 在计算视图中提供“Join”、“Union”、“Projection”以及“Aggregation”4 种工具，如图 8-86 所示。你能在红框标出位置找到它们。

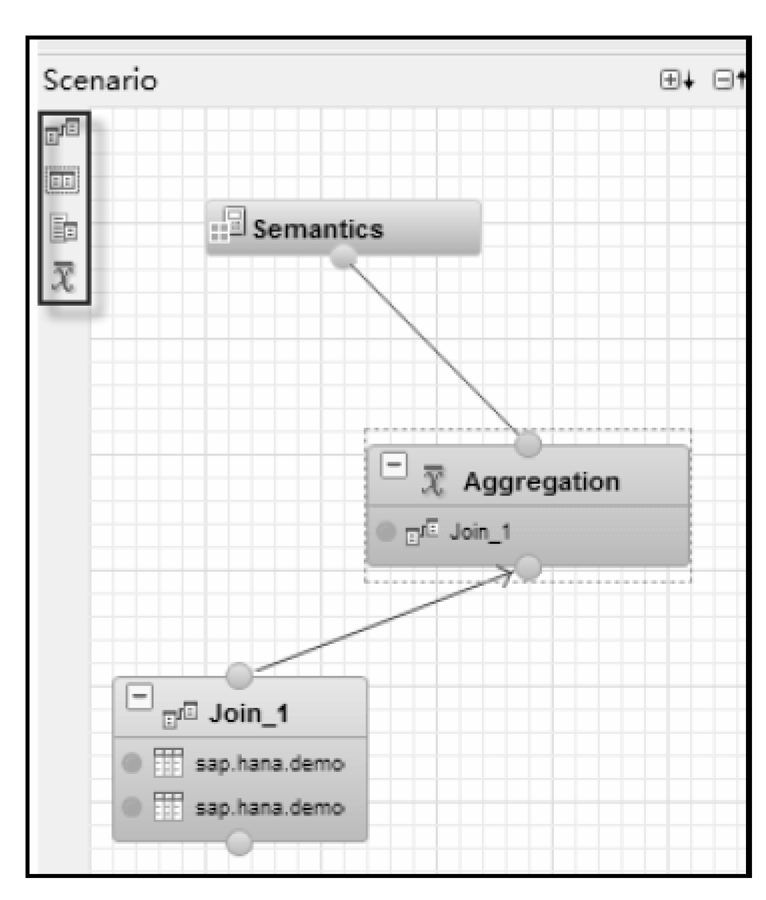


图 8-86

(3) 在本例中我们会使用“Join”和“Aggregation”两种工具。首先我们将需要“Join”的数据表或者属性/分析视图拖曳到新建的“Join”节点中去，如图 8-87 所示。

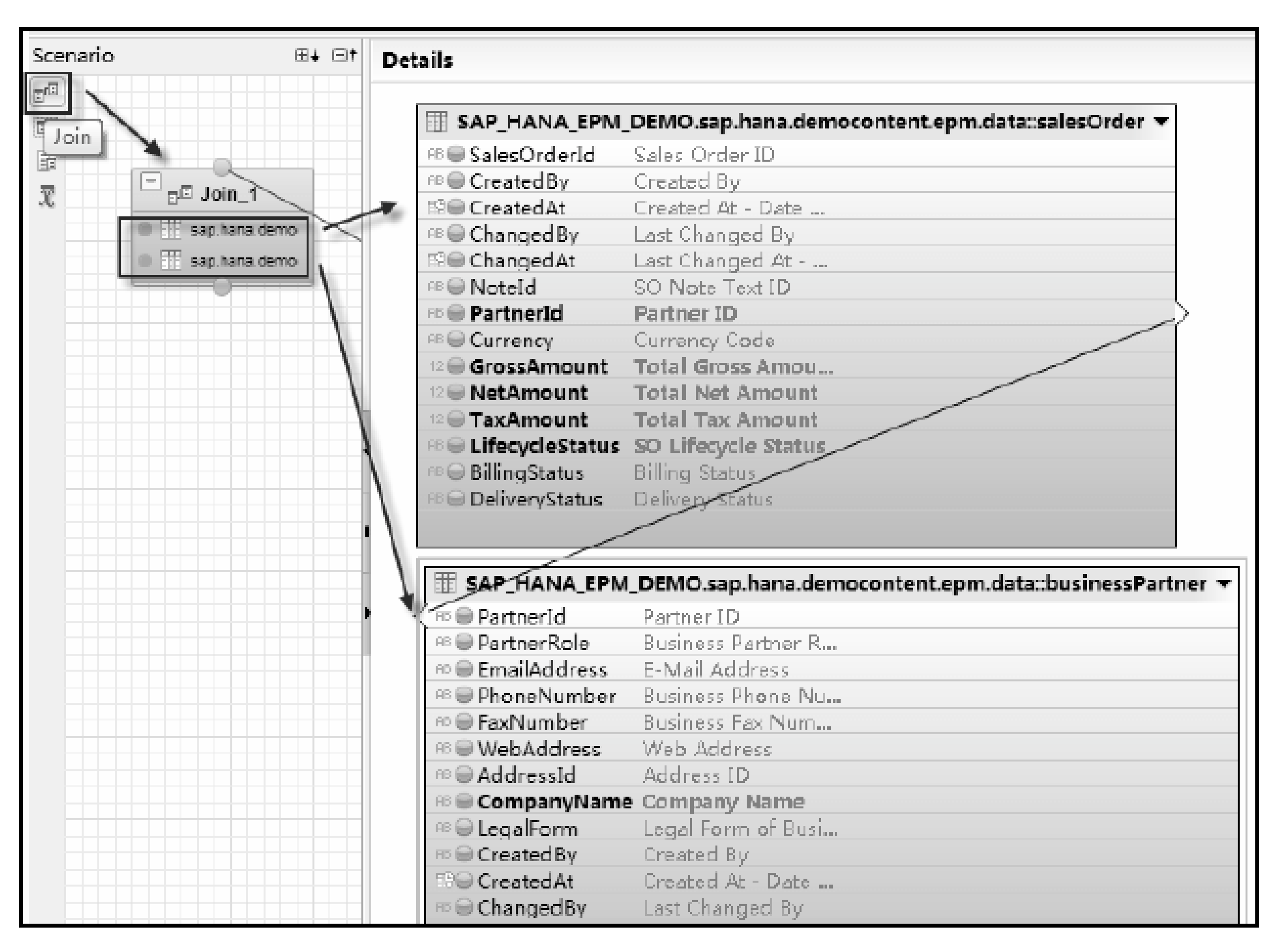


图 8-87



(4) 接下来我们在“Output”面板中定义输出列(见图 8-88)。

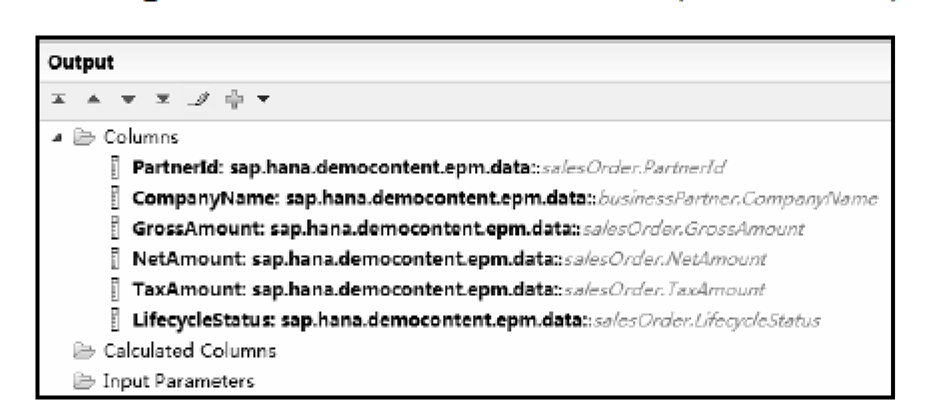


图 8-88

(5) 将配置好的“Join1”节点连接到“Aggregation”节点(见图 8-89)。

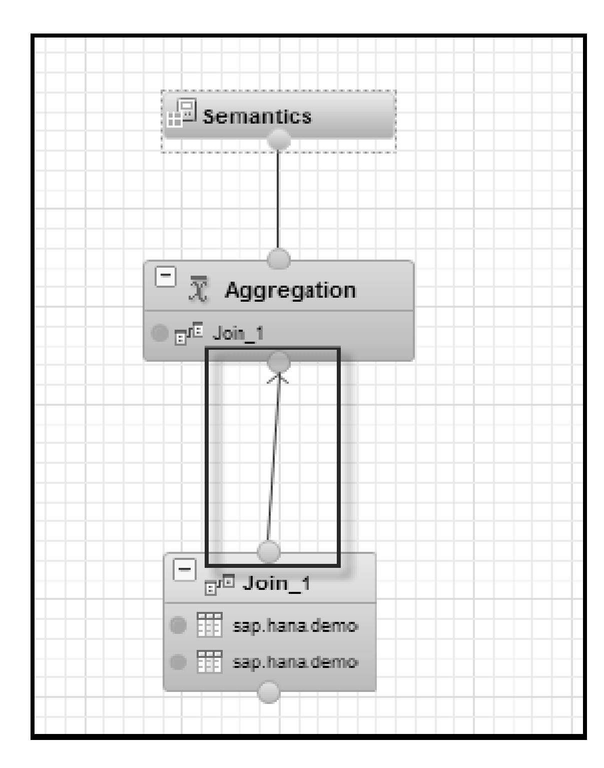


图 8-89

(6) 在“Aggregation”节点配置聚合列(见图 8-90)。

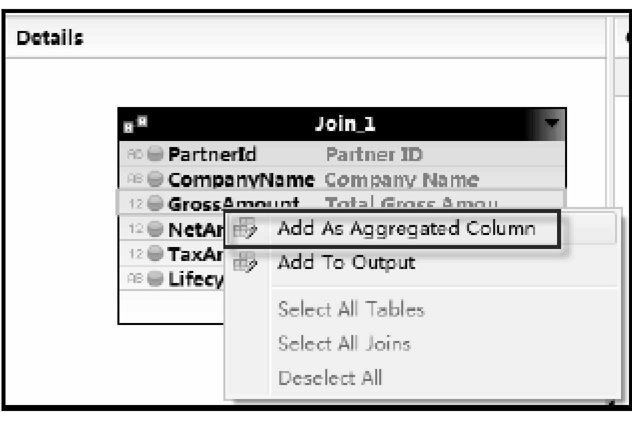


图 8-90

(7) 保存并激活计算视图(见图 8-91)。



图 8-91



(8) 预览结果。我们以“Company Name”为横坐标，其销售总额(GrossAmount)为纵坐标(见图 8-92)。

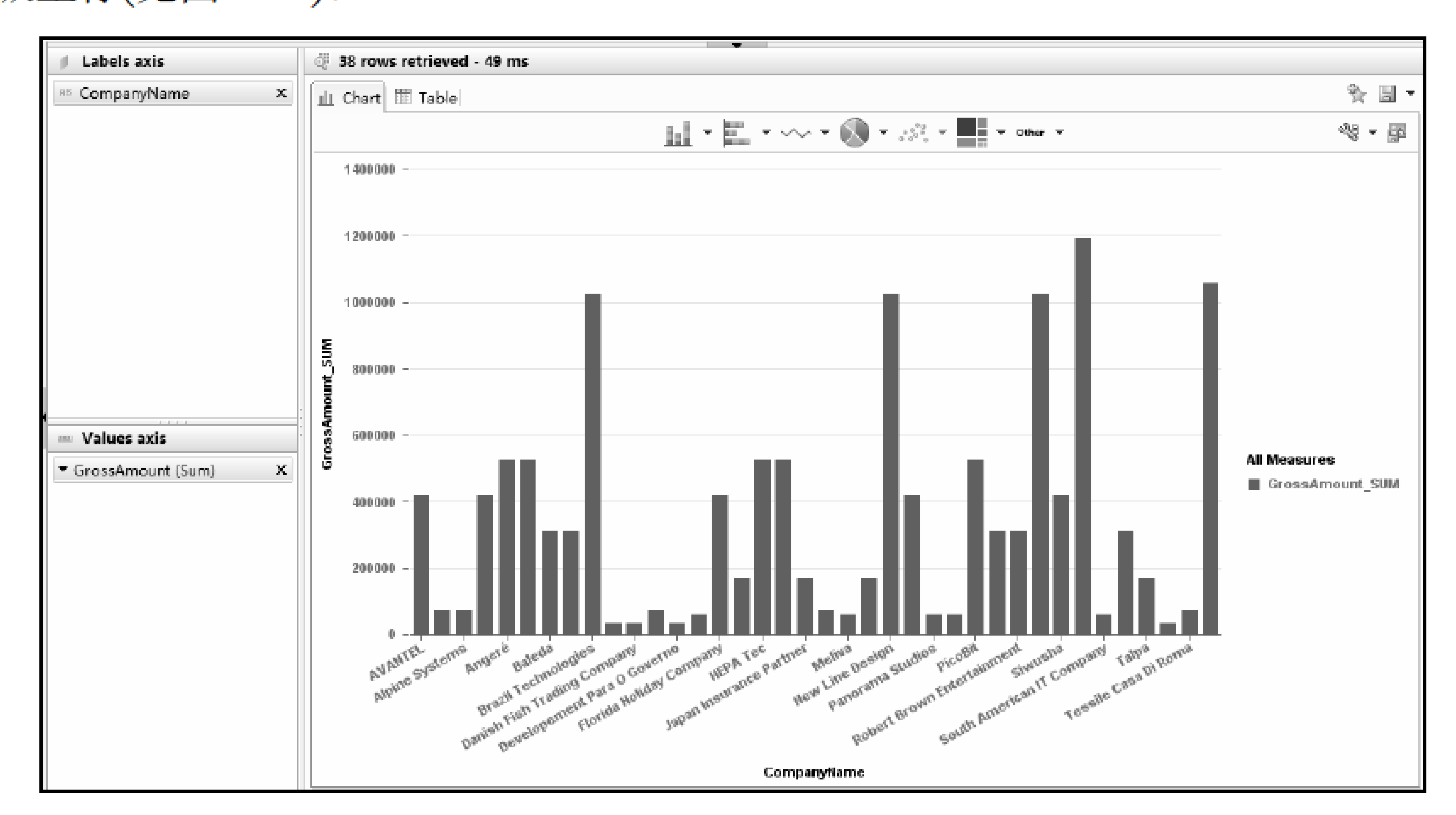


图 8-92

(9) 同样的，我们也能切换不同的图表来做进一步分析(见图 8-93)。

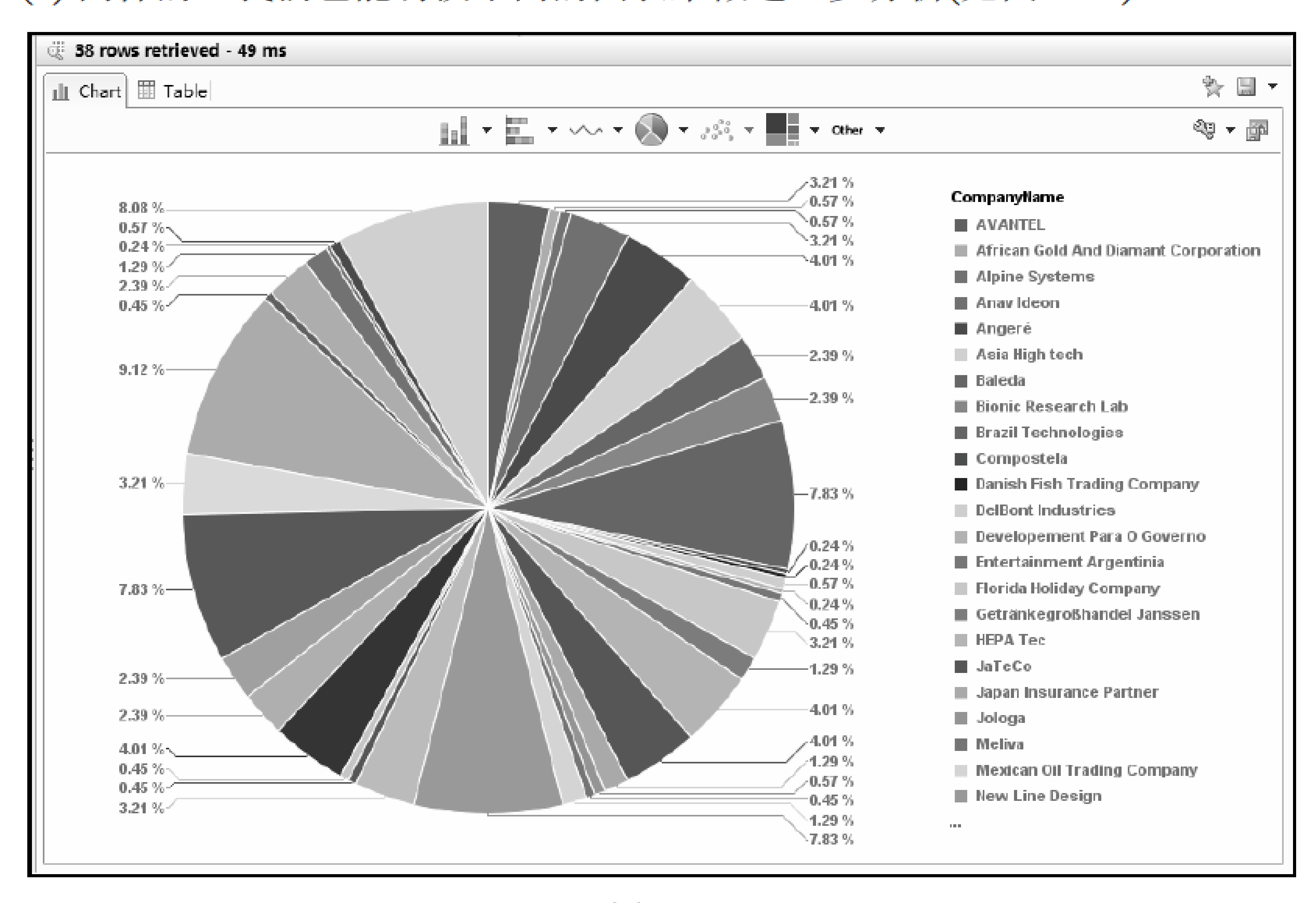


图 8-93

到这里，我们已经介绍完简单的计算视图的创建方法了。如果你觉得你已经掌握得差不多了，那就进入练习环节做些练习吧。

本章小结与练习

通过本章的学习，你应该已经掌握了如下的知识点：

- 属性数据和度量数据的概念；
- SAP HANA 视图的概念；
- 如何在 SAP HANA 中操作数据表；
- 什么是 SAP HANA Studio 的包，以及如何创建包；
- 属性视图、分析视图以及计算视图的区别。

练习

1. 在 SAP HANA Studio 中创建一个列存储的数据表。
2. 尝试用外部文件为你新创建的数据表导入内容。
3. 新建一个包。
4. 基于 SAP EPM 实例中的数据表，在新建的包里面创建属性视图，分析视图。
5. 创建一个计算视图，并做列聚合操作。



第九章　SAP HANA SQL 基础

第一节　SAP HANA SQL 简介

本章的目的在于帮助读者熟悉基本 SQL，为后续的 SAP HANA Script 学习打下基础。如果你对 SQL 已经非常熟悉，可以跳过本章前面内容，直接转到本章小结和练习，花几分钟温故知新，然后开始你的 SAP HANA SQL Script 之旅。

众所周知，SQL 是结构化查询语言(Structured Query Language)的缩写，是业界标准数据库查询语言。1986 年，美国国家标准学会(ANSI)对 SQL 进行规范，使其成为关系型数据库管理系统(relational database management systems，RDBMS)的标准语言。1987 年，SQL 在国际标准组织(ISO)的支持下成为国际标准。

说到关系型数据库，大家可能知道数据库实际上有很多种类型，除了熟知的关系型数据库，还有基于对象的(Object-oriented)数据库，基于 XML 的数据库类型等。因本文讨论基于 SAP HANA 平台的开发，这一章就仅讨论关系型数据库和标准 SQL。

市面上常见的关系型 DBMS 产品包括：

- Oracle 11g
- IBM DB2
- Microsoft SQL Server
- SAP HANA

说到关系型数据库管理系统，也借这个机会说明下几个常用的术语和缩写：数据库(DB)、数据库管理系统(DBMS)、数据库系统(DBS)以及数据库管理系统的三层架构(如图 9-1 所示)。

- 数据库(DB，Database)，其仅是结构化数据的集合。
- 数据库管理系统(DBMS，Database Management System)是用来管理数据库的大型软件系统，所有对数据库的增删改查都必须通过 DBMS 操作完成。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整

性。用户可以通过 DBMS 访问数据库中的数据，数据库管理员也是通过 DBMS 进行数据库的维护工作。

- 数据库系统(DBS，Database System)是某个特定的数据库再加上 DBMS，我们称之为数据库系统。

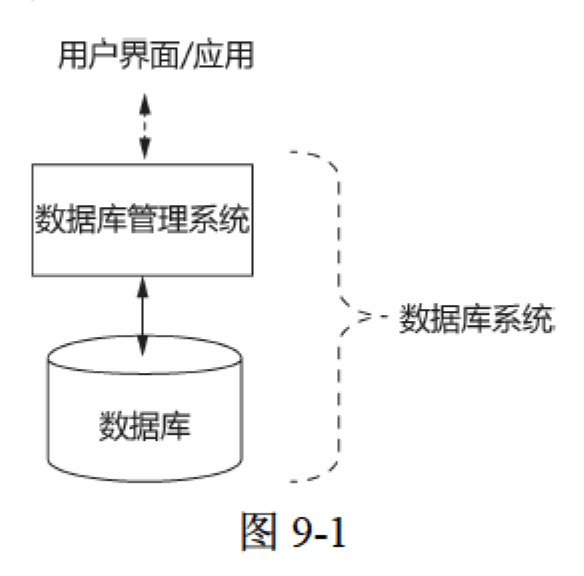


图 9-1

上面提到，DBMS 是一个大型的软件系统，这个系统的架构包含三个层次：外层、内层和中间作为桥梁的抽象层。一般用户看到的都是(External Schema)外部层次。分层的原因和其他大型软件系统类似，主要是增加数据独立性，避免各个层次的依赖。比如开发人员在内层对数据构造的修改，对外层(也就是用户)就不会产生不必要的影响。同理，终端用户也能够专注处理业务数据而不必关心数据在计算机中的物理存储和表示，如图 9-2 所示。

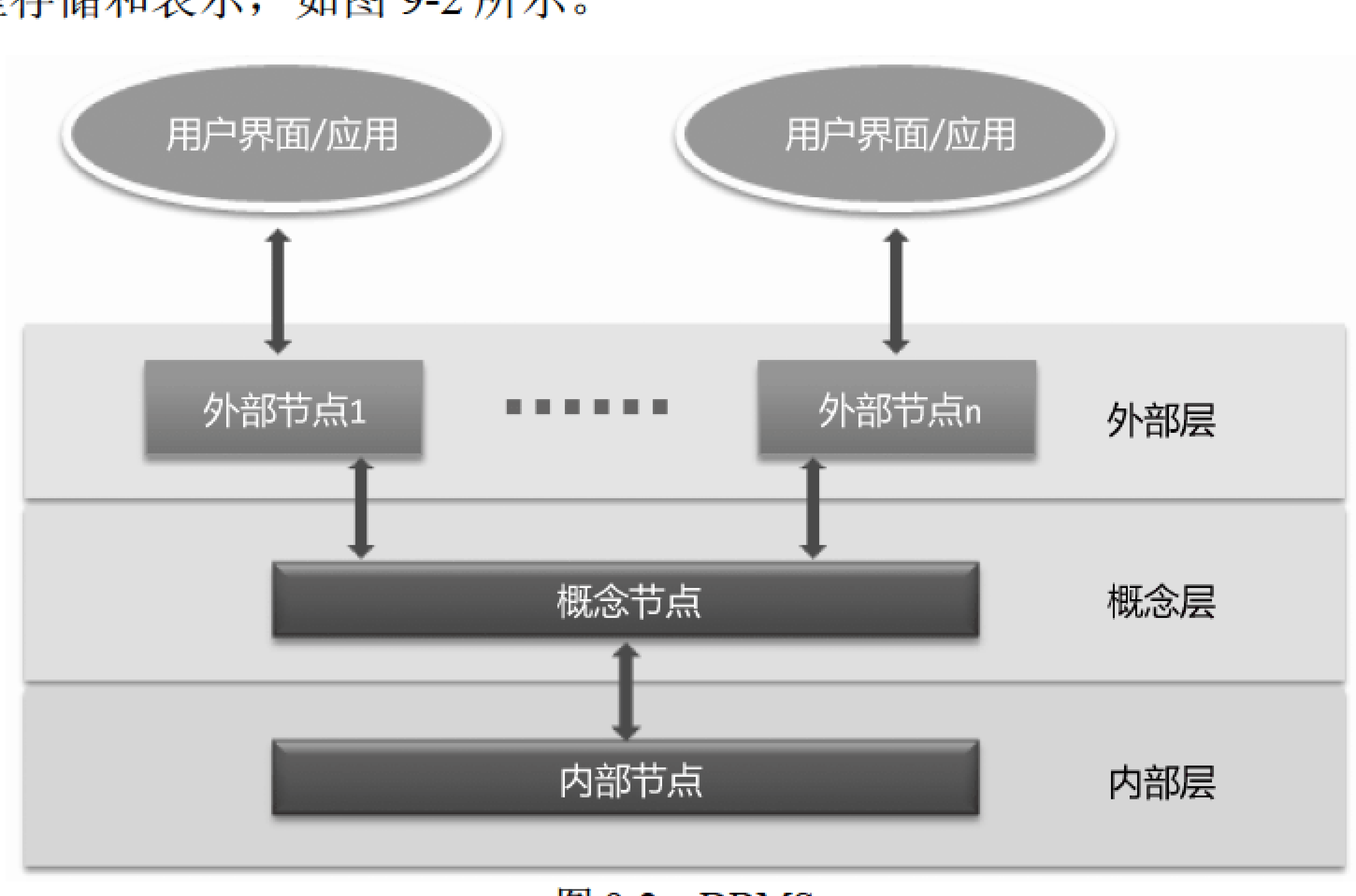


图 9-2　DBMS

针对图 9-2，这里我们再解释下这三个层次：外部层(External Level)，概念层(Conceptual Level)和内部层(Internal Level)。

- 内部层(也可以叫做物理层)是数据库最低一级的逻辑描述，它描述了数据在存储介质上的存储方式和物理结构。举个例子，表的实际存储文件，还有我们创建的索引等，在数据库都有一个实际的存储文件。对于一个数据库系统而言，物理级是客观存在的，它是进行数据库操作的基础，有数据，就有物理基本的数据文件。但一般用户，甚至有的应用开发人员不太会去关心物理层次。我们了解即可。
- 外部层是终端用户组所看到的数据库的数据视图。可以说外部层次是内部层次的一个子集，仅包含内部层次中允许特定用户使用的那部分数据。比如我们为某些特定用户组建立的视图就属于外部层次。
- 概念层是外部层和内部层的中间桥梁，是由数据库设计者从数据统一的角度构造的全局逻辑结构。我们常说的基本表就属于概念层。

这三个层次之间也有相应的映射关系。数据的独立性就是通过这个三个层次和层次间的映射关系实现。比如当中间层也就是概念层发生改变时(比如增加了一些新的表，对某些表增加了字段等)，用户看到的外模式(比如视图中的数据)可以保持不变。再打个比方，当数据的存储结构发生变化时，比如行存储改为列存储，减少了某些 INDEX，对最终用户看到视图或者应用而言，也没有任何的变化和影响。

了解完 SQL 和 DBMS，在我们正式进入 SAP HANA SQL 的语法前，我们先问个非常简单和基本的问题：

“为什么关系型数据库叫做‘关系型’数据库？”

这对于长期使用过程化开发语言或者面向对象的开发人员来说，刚接触非过程化语言 SQL 是个有益的提醒。

何谓关系(relation)？“关系”简单来说就是笛卡尔乘积的子集，如图 9-3 所示。而 SQL 是以数据记录(records)的合集(set)作为操纵对象。我们写 HANA SQL 或者 HANA SQL Script 必须先转换以前过程化编程语言，对变量和单条数据依次操作的思路，现在我们是通过命令直接操作数据集合。这也是为什么我们先提这个非常基础的“为什么关系型数据库叫做‘关系型’数据库”的问题了。

从另一个层面来说，“表”就是“关系”的展现形式。我们写 SQL，基本上就是对表这种形式数据集合的操作(见图 9-4)。



R ⊆ A x B x C

Relations

```
DNumber = {D035403, D040823, D054467, …}
Name = {Jim, Vishal, Werner, …}
BoardMemberSince = {2001, 2008, 2010, …}

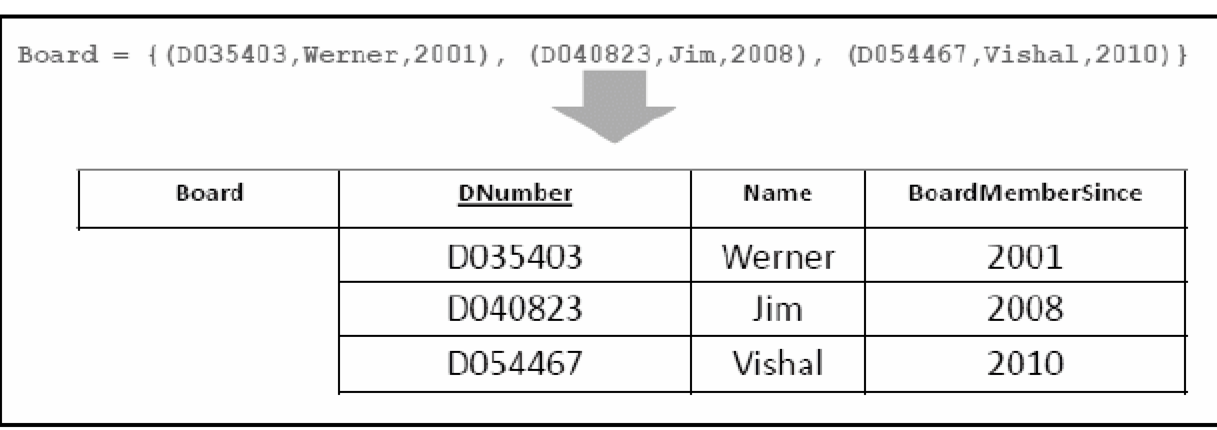
DNumber × Name × BoardMemberSince =
  {(D035403,Jim,2001),     (D035403,Jim,2008),     (D035403,Jim,2010),
   (D035403,Vishal,2001),  (D035403,Vishal,2008),  (D035403,Vishal,2010),
   (D035403,Werner,2001),  (D035403,Werner,2008),  (D035403,Werner,2010),
   (D040823,Jim,2001),     (D040823,Jim,2008),     (D040823,Jim,2010),
   (D040823,Vishal,2001),  (D040823,Vishal,2008),  (D040823,Vishal,2010),
   (D040823,Werner,2001),  (D040823,Werner,2008),  (D040823,Werner,2010),
   (D054467,Jim,2001),     (D054467,Jim,2008),     (D054467,Jim,2010),
   (D054467,Vishal,2001),  (D054467,Vishal,2008),  (D054467,Vishal,2010),
   (D054467,Werner,2001),  (D054467,Werner,2008),  (D054467,Werner,2010), …}

 Board ⊆ DNumber × Name × BoardMemberSince

 Board ⊆ { (D035403,Jim,2001),     (D035403,Jim,2008),     (D035403,Jim,2010)
           (D035403,Vishal,2001),  (D035403,Vishal,2008),  (D035403,Vishal,2010),
           (D035403,Werner,2001),  (D035403,Werner,2008),  (D035403,Werner,2010),
           (D040823,Jim,2001),     (D040823,Jim,2008),     (D040823,Jim,2010),
           (D040823,Vishal,2001),  (D040823,Vishal,2008),  (D040823,Vishal,2010),
           (D040823,Werner,2001),  (D040823,Werner,2008),  (D040823,Werner,2010),
           (D054467,Jim,2001),     (D054467,Jim,2008),     (D054467,Jim,2010),
           (D054467,Vishal,2001),  (D054467,Vishal,2008),  (D054467,Vishal,2010),
           (D054467,Werner,2001),  (D054467,Werner,2008),  (D054467,Werner,2010), …}

 Board = {(D035403,Werner,2001), (D040823,Jim,2008), (D054467,Vishal,2010)}
```

图 9-3



Board	DNumber	Name	BoardMemberSince
	D035403	Werner	2001
	D040823	Jim	2008
	D054467	Vishal	2010

图 9-4



如上所述，SQL 就是对数据集和关系的操作。相对其他语言，我们总结下 SQL 的基本特点：

- SQL 是命令式的语言，而非过程式，是描述“what”的问题而不是“how”。
- SQL 执行前先经过解析，优化，优化器会决定执行计划(Execution Plan)。

- SQL 面向数据集合，而非处理单个数据记录，使用 SQL 一次性就可以读取、修改、删除多个数据记录。
- SQL 是标准化的，不容许对语法的无管控地增加。

如果我们从功用角度对 SQL 语言分个类，SQL 主要包含三大部分：

- DDL-Data Definition Language(定义数据，比如创建表和视图等)
  - 例如 CREATE、ALTER、DROP、RENAME 等语句
- DML-Data Manipulation Language(操纵数据，即数据的增删改查)
  - 例如 SELECT、INSERT、UPDATE、DELETE 等语句。
- DCL-Data Control Language(控制数据，权限相关)
  - 例如 GRANT、REVOKE 等语句。

图 9-5 是常见的 DML SQL 的一个例子：

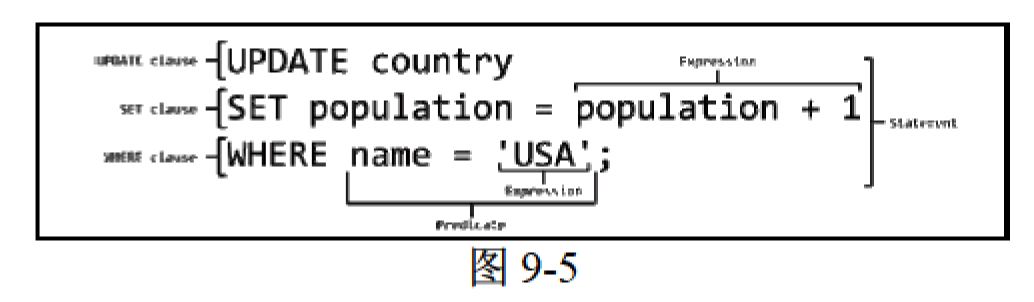


图 9-5

这里需要指出 SQL 的三个基本术语，后续章节讲解时我们可能用到。

- 谓词(Predicate)
- 操作符(Operator)
- 表达式(Expression)

谓词是由一个或多个表达式和运算符构成、返回 TRUE、FALSE 或者 UNKNOWN。常见的谓词如表 9-1 所示。

表 9-1

分　类	说　明
比较谓词 (Comparison Predicates)	<expression> { = | != | <> | > | < | >= | <= } [ANY | SOME| ALL] { <expression_list> | <subquery> }
范围谓词 (Range Predicate)	<expression1> [NOT] BETWEEN <expression2> AND <expression3>
In 谓词 (In Predicate)	<expression> [NOT] IN { <expression_list> | <subquery> }

(续表)

分　类	说　明
Exists 谓词 (Exists Predicate)	[NT] EXISTS (<subquery>)
LIKE 谓词 (LIKE Predicate)	<expression1> [NOT] LIKE <expression2> [ESCAPE <expression3>]
NULL 谓词 (NULL Predicate)	<expression> IS [NOT] NULL

下面是使用了比较、范围、IN、LIKE 和 NULL 的一个例子，仅作语法示例。

```
SELECT TOP 10 DISTINCT CARRID, CONNID, FLDATE, PRICE, CURRENCY
    FROM SFLIGHT
    WHERE CARRID = 'LH'
      AND PRICE <1000
      AND FLDATE BETWEEN 20110101 AND 20111231
      AND CONNID IN ('3577', '3517')
      AND PLANETYPE LIKE '747%'
      AND SEATSMAX IS NOT NULL
    ORDER BY PRICE ASC
```

操作符(Operator)用于计算、值的比较或者赋值。从功能的角度划分，包含表 9-2 所示的几类。

表 9-2

分　类	说　明
算术操作符	如加法、减法、乘法和除法
字符串操作符	<expres s i on> || <expres s i on> 级联操作符结合两项类似字符串、表达式或者常量到一项中
比较操作符	= , > , < , <> 等
逻辑操作符	AND , OR , NOT
集合操作符	UION , UION ALL , INTERSECT , EXCEPT

表达式(Expression)有如表 9-3 所示的几类。



表 9-3

分　类	说　明
CASE 表达式	CASE... WHEN THEN ..END
Function 表达式	SQL built-in functions
聚合表达式	COUNT，SUM，AVG，MIN/MAX 等
子查询表达式	包含在()中的 SELECT 子语句

在后面数据操纵 DML 章节中，我们会对上面的这些概念举例讲解实际应用。

可能有人会问在 SAP HANA SQL 中的注释是如何标注的，这里顺便指出，对于单行可以使用“–”，多行可以使用“/* comment */ ”，例如：

- 单行注释

  -- <注释部分>
- 多行注释

  /*

  <注释部分>

  */

了解了 SQL 的基本概念后(如 DDL/DML/DCL，谓词，操作符，表达式)，现在我们开始细节系统地回顾一下 SAP HANA SQL 的技术点，先从数据定义 DDL 开始。

第二节　定义数据 DDL

一、数据类型

简单来说，数据用来储存谁(who)、何时(when)、在哪里(where)、做了什么(what，包含度量)。一般我们用字符表示人和物，日期和时间表示何时，度量多少用数字类型表示。9-4 表包含了 SAP HANA 的主要数据类型分类。

表 9-4

分　类	数 据 类 型
日期时间类型(Date time)	DATE，TIME，SECONDDATE，TIMESTAMP
数字类型 (Numeric)	TINYINT，SMALLINT，INTEGER，BIGINT，SMALLDECIMAL，DECIMAL，REAL，DOUBLE

(续表)

分 类	数 据 类 型
字符串类型 (Character String)	VARCHAR，NVARCHAR，ALPHANUM，SHORTTEXT
二进制类型(Binary)	VARBINARY
大数据对象类型(Large Object)	BLOB，CLOB，NCLOB，TEXT

我们分小类看一些实际的数据格式和一些例子。

- 日期时间类型(Date Time)

表 9-5 为日期格式，表 9-6 为时间格式。

表 9-5

格 式	描 述	例子和说明
YYYY-MM-DD	缺省格式	INSERT INTO my_tbl VALUES ('1957-06-13');
YYYY/MM/DD YYYY/MM-DD YYYY-MM/DD	YYYY from 0001 to 9999， MM from 1 to 12, DD from 1 to 31.	INSERT INTO my_tbl VALUES ('1957-06-13'); INSERT INTO my_tbl VALUES ('1957/06/13'); INSERT INTO my_tbl VALUES ('1957/06-13'); INSERT INTO my_tbl VALUES ('1957-06/13');
YYYYMMDD	ABAP DATS 数据格式	INSERT INTO my_tbl VALUES ('19570613');
MON	JAN. ~ DEC. 缩写	INSERT INTO my_tbl VALUES (TO_DATE('2040-Jan-10'，'YYYY-MON-DD')); INSERT INTO my_tbl VALUES (TO_DATE('Jan-10'，'MON-DD'));
MONTH	JANUARY – DECEMBER 月份名称	INSERT INTO my_tbl VALUES (TO_DATE('2040-January-10'，'YYYY-MONTHDD')); INSERT INTO my_tbl VALUES (TO_DATE('January-10'，'MONTH-DD'));
RM	Roman numeral month (I-XII; JAN = I).	INSERT INTO my_tbl VALUES (TO_DATE('2040-I-10'，'YYYY-RM-DD')); INSERT INTO my_tbl VALUES (TO_DATE('I-10'，'RM-DD'));
DDD	Day of year (1-366).	INSERT INTO my_tbl VALUES (TO_DATE('204'，'DDD')); INSERT INTO my_tbl VALUES (TO_DATE('2001-204','YYYY-DDD'));
HH24:MI:SS	缺省格式 HH from 0 to 23. MI from 0 to 59. SS from 0 to 59.	INSERT INTO my_tbl VALUES ('23:59:59');



（续表）

格　　式	描　　述	例子和说明
HH:MI[:SS][AM|PM] HH12:MI[:SS][AM|PM] HH24:MI[:SS]	使用 AM/PM， 12/24 制式	NSERT INTO my_tbl VALUES ('3:59:59 PM'); INSERT INTO my_tbl VALUES ('3:47:39 AM'); INSERT INTO my_tbl VALUES ('9:9:9 AM'); INSERT INTO my_tbl VALUES (TO_TIME('11:59:59','HH12:MI:SS PM');
SSSSS	午夜后的总计秒数	INSERT INTO my_tbl VALUES (TO_TIME('12345'，'SSSSS'));

表 9-6

格　　式	描　　述	例子和说明
YYYY-MM-DD HH24:MI:SS.FF7	缺省格式	INSERT INTO my_tbl VALUES (TO_TIMESTAMP('2011-05-11 12:59.999','YYYY-MM-DD HH:SS.FF3'));
FF [1..7]	可设定秒的精度	

相应的日期时间转换功能见官方帮助文件，就不赘述了。

- 数字类型(Numeric)如表 9-7 所示。

表 9-7

格　　式	描　　述	例子和说明
TINYINT	8-bit unsigned integer	0 ～ 255
SMALLINT	16-bit signed integer	-32,768 ～ 32,767
INTEGER	32-bit signed integer	-2,147,483,648 ～ 2,147,483,647
BIGINT	64-bit signed integer	-9,223,372,036,854,775,808 ～9,223,372,036,854,775,807
DECIMAL(precision，scale)		0.0000259 DECIMAL(3,7) 1.0000259 DECIMAL(8,7) 123.4567 DECIMAL(7,4) 如果是 DECIMAL(5，4), 3.14，3.1415,3.141592 存储后变为 3.1400，3.1415，3.1416。
SMALLDECIMAL	precision 1～16 Scale 369～368	比如 可以是 3.14，也可以是 3.1415、3.141592， 仅列存储支持 SMALLDECIMAL。
REAL	single-precision 32-bit floating-point number	0.4 在列存储的 REAL 数据为 0.400000005960464 5， 10*(0.04/0.2) 的 REAL 值为 1.9999998807907104 (不是 2)， 所以在用 WHERE or JOIN 的时候不要使用 REAL。

(续表)

格　式	描　述	例子和说明
DOUBLE	single-precision 64-bit	和 REAL 类似，不过是 64 bit
FLOAT(n)	N 从 1 到 53 变化。 如果 n 小于 25，就变成 32-bit REAL 如果大于等于 25. 就变成 64-bit DOUBLE。	浮点数 n 可以变化

- 字符串类型(Character String)如表 9-8 所示。

表 9-8

格　式	描　述	例子和说明
VARCHAR	ASCII 类型字符串 1 到 5000 长度	HANA
NVARCHAR	Unicode 编码字符串 1 到 5000 长度	HANA
ALPHANUM	32-bit 正负字符型整数 1 到 127 位	ALPHANUM (3) 101
SHORTTEXT	长度不定字符串 支持 text search	SAP HANA Rocks

- 二进制类型(Binary)如表 9-9 所示。

表 9-9

格　式	描　述	例子和说明
VARBINARY	二进制类型 长度 1 到 5000	X'00abcd'

- 大数据对象类型(Large Object)如表 9-10 所示。

表 9-10

格　式	描　述	例子和说明
BLOB	用于储存较大文件，如大文本或者图片，最大 2GB。	LOB 有一定的局限性: LOB columns 不能用做 primary key； LOB columns 不能使用 ORDER BY ，GROUP BY JOIN，UNION。
CLOB	大型 ASCII 字符类型数据	
NCLOB	大型 Unicode 字符类型数据	
TEXT	支持 textsearch 的大型 Unicode 字符类型数据	如果 SELECT TEXT 类型，实际是返回 NCLOB 型数据。



- 常量(Constant)如表 9-11 所示。

表 9-11

描　　述	例子和说明
字符串常量	'Brian'　'100'　N'abc (Unicode string)
数字常量	123 123.4
日期常量	'2010-01-01'
时间常量	'11:00:00.001'
时间翟义常量	'2011-12-31 23:59:59'
二进制字符串常量	X'00abcd'

现在我们大概了解了创建定义数据的“原料”，接下来看看如何用“原料”在 SAP HANA 里创建数据。本章主要介绍数据表和视图。

上一章我们使用了 EPM 模型作为建模案例，这里我们先用大家最熟悉的 SFLIGHT 来熟悉语法，在这章的结尾我们会做一个 SQL 要点小结，再使用 EPM 用例做一些练习。

创建节点(Schema)、表、视图有多种方式，这里我们先介绍简单直接地在 SQL CONSOLE 中通过 SQL 语法创建的方式。另外在后续的 HANA Native 开发中我们可以在 Project Explore 中通过创建特殊后缀 File 的方式创建 Schema、表、视图等。本章仅介绍前者。

二、创建列存储数据表

创建表的语法如下：

```
CREATE COLUMN TABLE TableName
(ColumnName1 Data Type,
 ColumnName2 Data Type,
 ColumnName3 Data Type NOT NULL,
 ColumnName4 Data Type NOT NULL,
 ColumnName5 Data Type DEFAULT Default-Value,
 ColumnName6 Data Type NOT NULL DEFAULT Default-Value,
PRIMARY KEY (ColumnName1, ColumnName2));
```

语法上 Primary key 不是必需的，因为 column table 容许存在相同行项目。

一般系统管理员为你创建用户的时候，和你的用户名相同的 Schema 也会同时被系统自动创建。当你用你的用户创建新表时，他们会缺省放到这个和你用户同名的 Schema 当中。我们也可以创建新的 Schema，这里用大家熟悉的 SFLIGHT 为例，我们创建一个新的 SCHEMA SFLIGHT。

```
CREATE SCHEMA SFLIGTH;
SET SCHEMA  SFLIGTH;
CREATE COLUMN TABLE SFLIGHT
     (MANDT  NVARCHAR(3) DEFAULT '000' NOT NULL ,
     CARRID NVARCHAR(3) DEFAULT '' NOT NULL ,
     CONNID NVARCHAR(4) DEFAULT '0000' NOT NULL ,
     FLDATE NVARCHAR(8) DEFAULT '00000000' NOT NULL ,
     PRICE  DECIMAL(15,2) DEFAULT 0 NOT NULL ,
     CURRENCY  NVARCHAR(5) DEFAULT '' NOT NULL ,
     PLANETYPE NVARCHAR(10) DEFAULT '' NOT NULL ,
     SEATSMAX  INTEGER DEFAULT 0 NOT NULL ,
     SEATSOCC  INTEGER DEFAULT 0 NOT NULL ,
     PAYMENTSUM DECIMAL(17,2) DEFAULT 0 NOT NULL ,
     SEATSMAX_B INTEGER DEFAULT 0 NOT NULL ,
     SEATSOCC_B INTEGER DEFAULT 0 NOT NULL ,
     SEATSMAX_F INTEGER DEFAULT 0 NOT NULL ,
     SEATSOCC_F INTEGER DEFAULT 0 NOT NULL ,
     PRIMARY KEY (MANDT,CARRID,CONNID,FLDATE));
```

- PRIMARY KEY

对于 PRIMARY KEY 字段，系统默认这些字段为 NOT NULL，可以不用显式的加上 NOT NULL。



- DEFAULT

使用 DEFAULT，如果在 INSERT 时没有提供字段值时，缺省值会被使用。

- UNIQUE

如果表中的某个字段的值需要是唯一的，我们可以使用 UNIQUE 指明。比如车牌照字段就是好的例子。

```
CREATE COLUMN TABLE CAR
      (CARID VARCHAR(3) PRIMARY KEY,
    PLATENUMBER VARCHAR(10) UNIQUE NOT NULL,
    BRAND VARCHAR(20),
    COLOR VARCHAR(10),
    HP INTEGER,
    OWNER VARCHAR(3));
```

- 双引号

在创建 Schema 或数据表时，其名称也可也用双引号标示，比如“Table_name”，如需显式的使用小写字符或空格。

例如：

```
CREATE SCHEMA "Sflight 1";
```

但我们不建议在表名中使用空格或大小写混用，很容易引起混淆。比如下面较极端的例子：

```
CREATE COLUMN TABLE "Don’t do it"
("sap" INTEGER,
 "saP" INTEGER,
 "sAp" INTEGER,
 "SAP" INTEGER);
```

除了创建 column table，我们也可以根据需要创建其他类型的表，比如 row table，支持 Time Travel 的 history column table，以 Session 为生命周期的 global temporary table、local temporary table 等，这里就不一一展开。

三、数据表操作

本部分我们来看看如何在 SAP HANA Studio 中对数据表进行操作。在所有数据表操作中，对数据表的修改最为重要，因此我们首先来介绍数据表修改操作，即 ALTER TABLE。

通过 ALTER TABLE，我们可以修改表的属性，如增减 PRIMARY KEY，增加列，减少列，修改列的属性，NOT NUUL，UNIQUE 的属性，有没有 DEFAULT VALUE 等。

- 增加列

```
ALTER TABLE SFLIGHT.SFLIGHT
ADD (NEWCOLUMN1 INTEGER,
                 NEWCOLUMN2 INTEGER NOT NULL DEFAULT 30);
```

如果是 NOT NULL，需要给相应字段 DEFAULT 值，数据表插入数据时这些 DEFAULT 值会被用到。

- 减少列

```
ALTER TABLE SFLIGHT.SFLIGHT
```

```
DROP (NEWCOLUMN2);
```

- 修改列

```
ALTER TABLE SFLIGHT.SFLIGHT
ADD (NEWCOLUMN3 VARCHAR(20));
ALTER TABLE SFLIGHT.SFLIGHT
ALTER (NEWCOLUMN3 VARCHAR(30) NOT NULL DEFAULT 'TEST');
```

修改列的数据类型时要注意数据类型的兼容性。上面例子修改列类型从VARCHAR 20到VARCHAR 30没有问题，但如果尝试从VARCHAR(5)转到DECIMAL就会报错，长度从20位转到较短的10位也不允许，因为会产生已有数据类型兼容和精度丢失的问题。

为了验证这些信息，首先我们在SQL控制台输入如下语句来测试20位转到10位的尝试：

```
ALTER TABLE SFLIGHT.SFLIGHT
ALTER (NEWCOLUMN3 VARCHAR(10) NOT NULL DEFAULT 'TEST');
```

我们会得到如下的错误信息：

```
Could not execute 'ALTER TABLE SFLIGHT.SFLIGHT ALTER (NEWCOLUMN3
VARCHAR(10) NOT NULL DEFAULT 'TEST')'
SAP DBTech JDBC: [7] (at 35): feature not supported: cannot shorten
the field length: NEWCOLUMN3: line 2 col 8 (at pos 35)
```

接下来我们来进行VARCHAR(20)到DECIMAL(10，3) 的尝试

```
ALTER TABLE SFLIGHT.SFLIGHT
ALTER (NEWCOLUMN3 DECIMAL(10,3) NOT NULL DEFAULT 9.888);
```

我们能得到如下的错误信息：

```
Could not execute 'ALTER TABLE SFLIGHT.SFLIGHT ALTER (NEWCOLUMN3
DECIMAL(10,3) NOT NULL DEFAULT 9.888)'
SAP DBTech JDBC: [7] (at 35): feature not supported: cannot shorten
the field length: NEWCOLUMN3: line 2 col 8 (at pos 35)
```

- 修改PRIMARY KEY

```
ALTER TABLE SFLIGHT.SFLIGHT
DROP PRIMARY KEY;

ALTER TABLE SFLIGHT.SFLIGHT
ADD PRIMARY KEY (MANDT,CARRID,CONNID,FLDATE,NEWCOLUMN1);
```

- 重命名数据表 RENAME TABLE

```
RENAME TABLE SFLIGHT.SFLIGHT TO SFLIGHT2;
```

- 废除数据表 DROP TABLE

```
DROP TABLE SFLIGHT.SFLIGHT;
```

- 重命名数据表列 RENAME COLUMN

```
RENAME COLUMN SFLIGHT.SFLIGHT.NEWCOLUMN3 TO NEWCOLUMN9;
```

到这里，我们已经了解了创建表的基础，基本表创建好后，我们可以通过上载 CSV 文件的方式为表填充数据。通过文件上载在数据供应章节有说明，本章就略过。

第三节 操作数据 DML

现在我们定义好了数据结构，并给这些基本表填充了测试数据，接下来我们看看操作数据的部分。数据操作主要包括：

- 数据读取(从单表和从多表中读取数据)
- 数据更新(增、删、改)

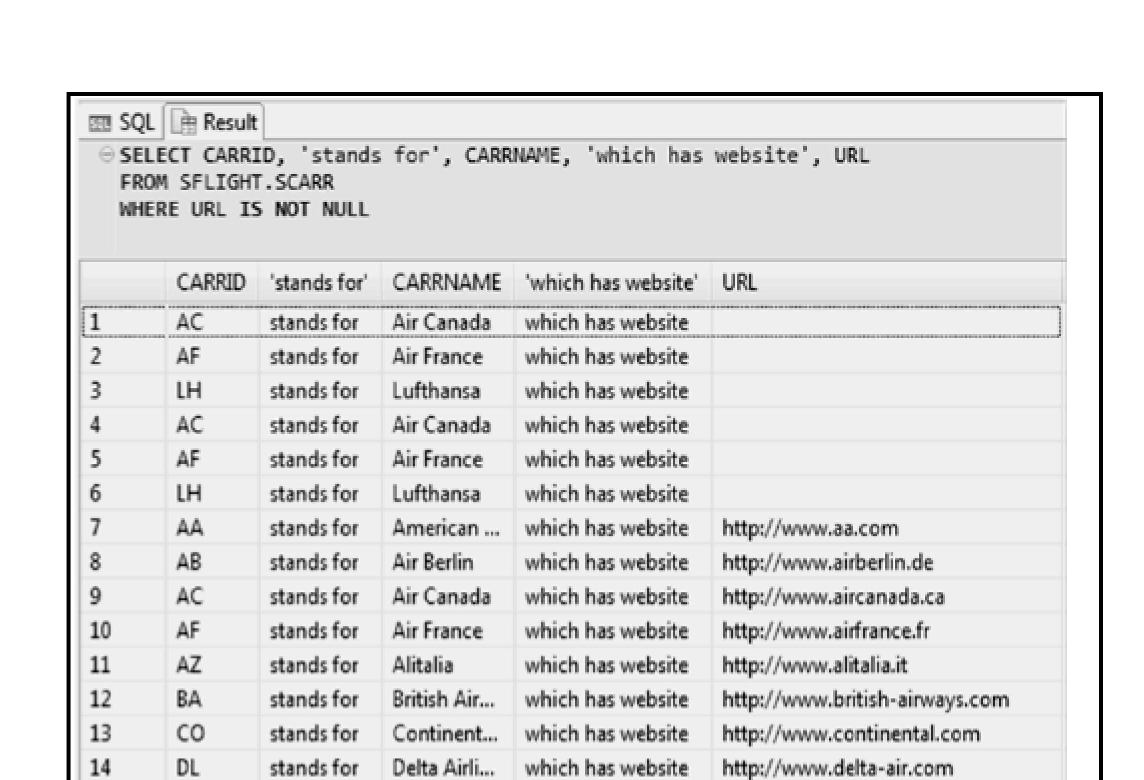
一、SELECT——从单表或视图读取数据

SELECT 是 SAP HANA SQL 中最基本，也是最重要的数据读取语句。基本的 SELECT 读取语句如下：

```
SELECT CARRID, CONNID, FLDATE, BOOKID, CUSTOMID, ORDER_DATE, PASSNAME,
AGENCYNUM
FROM SFLIGHT.SBOOK
WHERE CLASS = 'Y'
ORDER BY ORDER_DATE DESC
```

需要时我们可以人为地在 Project List 中增加一些列(见图 9-6)，例如：

```
SELECT CARRID, 'stands for', CARRNAME, 'which has website', URL
FROM SFLIGHT.SCARR
WHERE URL IS NOT NULL
```


SQL　Result

```
SELECT CARRID, 'stands for', CARRNAME, 'which has website', URL
FROM SFLIGHT.SCARR
WHERE URL IS NOT NULL
```

	CARRID	'stands for'	CARRNAME	'which has website'	URL
1	AC	stands for	Air Canada	which has website	
2	AF	stands for	Air France	which has website	
3	LH	stands for	Lufthansa	which has website	
4	AC	stands for	Air Canada	which has website	
5	AF	stands for	Air France	which has website	
6	LH	stands for	Lufthansa	which has website	
7	AA	stands for	American ...	which has website	http://www.aa.com
8	AB	stands for	Air Berlin	which has website	http://www.airberlin.de
9	AC	stands for	Air Canada	which has website	http://www.aircanada.ca
10	AF	stands for	Air France	which has website	http://www.airfrance.fr
11	AZ	stands for	Alitalia	which has website	http://www.alitalia.it
12	BA	stands for	British Air...	which has website	http://www.british-airways.com
13	CO	stands for	Continent...	which has website	http://www.continental.com
14	DL	stands for	Delta Airli...	which has website	http://www.delta-air.com

图 9-6

这里顺便指出，NULL 值是 SQL 中一个特别的预留字，和字符串中的空格是不等同的。后续章节对 NULL 有相应说明。

回到简单的 SELECT 操作，我们也可以在 SELECT 的时候为 Project List 中的一些列做一些简单的计算。

```
SELECT CARRID, CONNID, FLDATE, PRICE*0.8 AS DISCOUNTED
FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'LH'
```

二、SELECT——从多个表和视图中读取数据

前面的章节我们了解了如何从单表读取数据，现在我们延伸到如何从多个表和视图中读取数据。基本上，要从多个表中读取数据有三种方式。

- UNION

使用 UNION 我们可以把不同的查询输出结果合并成一个结果，输出的列的名称以第一个 SELECT 语句为准。可以合并的前提是首先需要合并的查询结果的列的数量必须一致，其次需要合并的查询结果的列的数据类型必须兼容。

结果集合并后可能包含一些重复的项，如果不需要包含重复项，仅使用 UNION 即可。但 UNION 的执行性能会比较差。

实例：

```
SELECT CARRID, CONNID, FLDATE FROM SFLIGHT.SBOOK
WHERE CLASS = 'Y'
UNION ALL
SELECT CARRID, CONNID, FLDATE FROM SFLIGHT.STICKET
```

运行结果如图 9-7 所示。

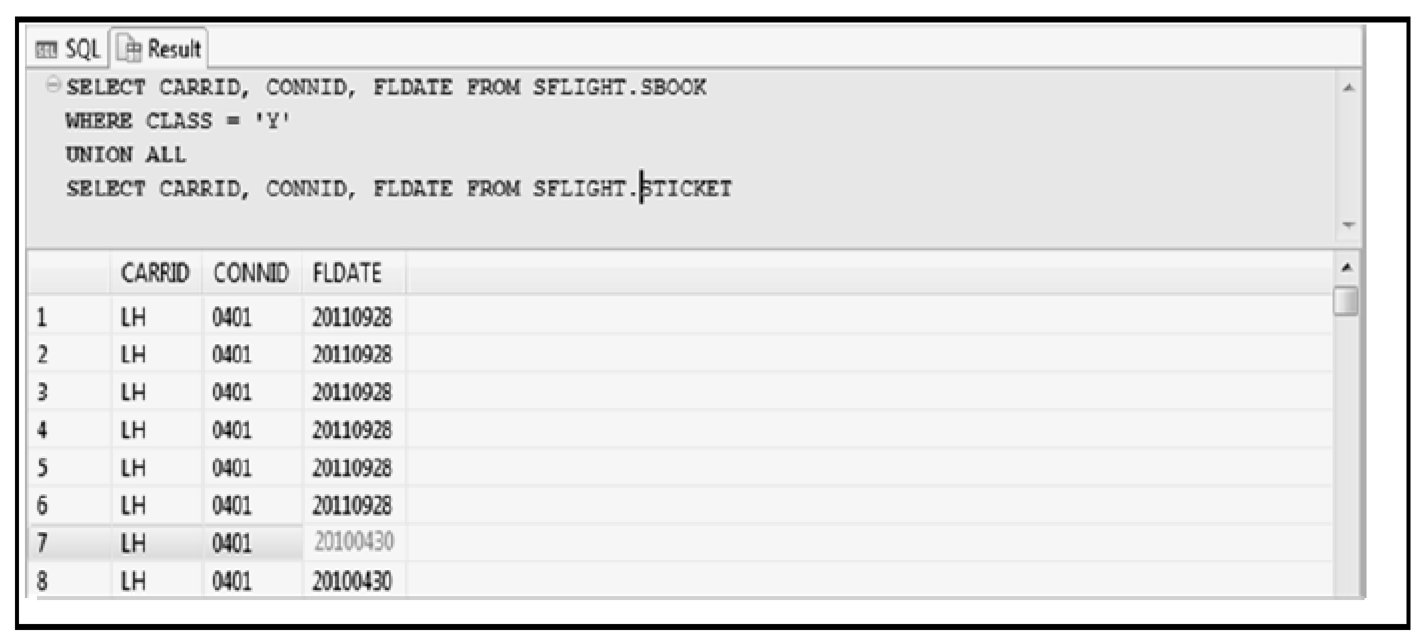


图 9-7

- JOIN

交叉连接(Cross Join)：

对于交叉连接方式来说，由于其不能设置连接条件，因此会返回连接表的所有行。比如 L 表含有 10 行数据，Y 表含有 20 行数据，那么 L 表和 R 表交叉连接的结果集就是 10×20＝200 行的结果集，如图 9-8 所示。

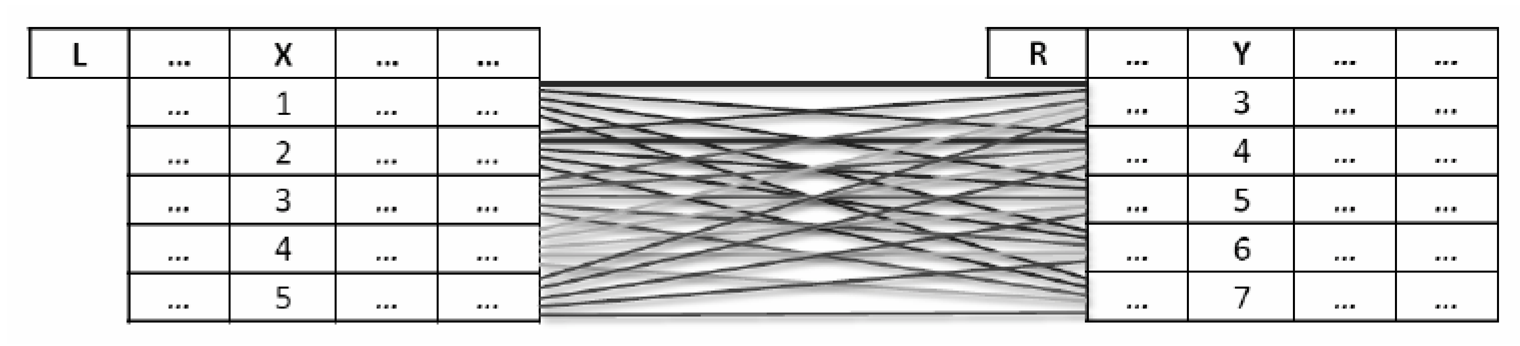


图 9-8

实例：

```
SELECT PLANETYPE, SEATSOCC, BOOKID, ORDER_DATE
FROM SFLIGHT.SFLIGHT CROSS JOIN SFLIGHT.SBOOK
```

运行结果如图 9-9 所示。

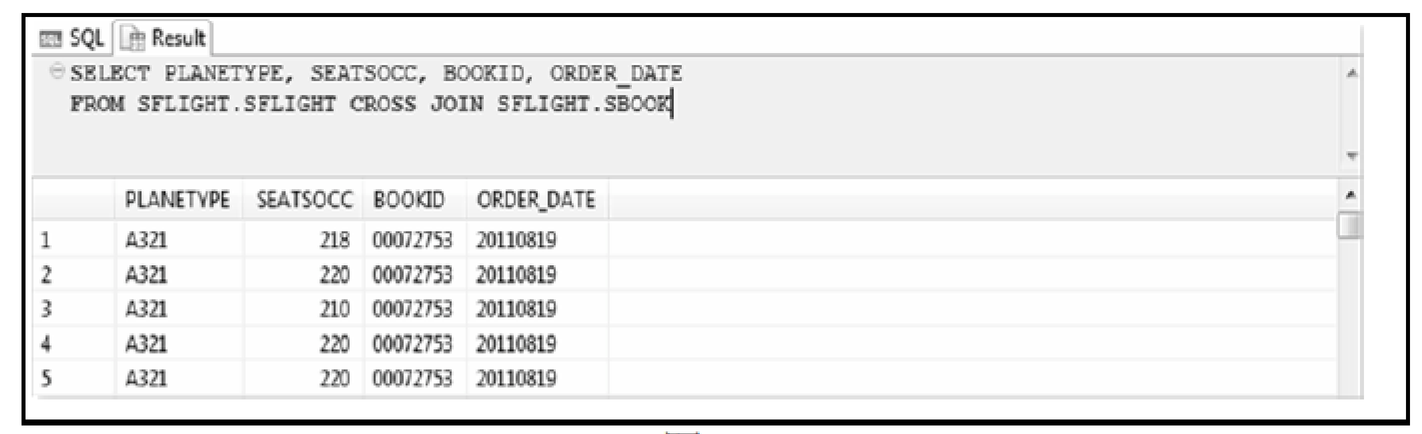


图 9-9

内连接(Inner Join)：

内连接方式下，结果集只返回所连接表中满足连接条件的数据行，如图 9-10 所示。

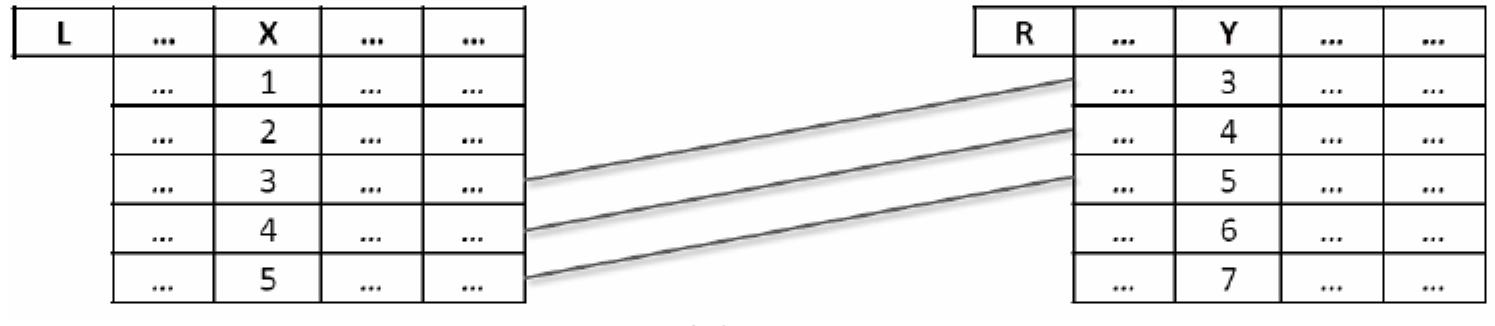
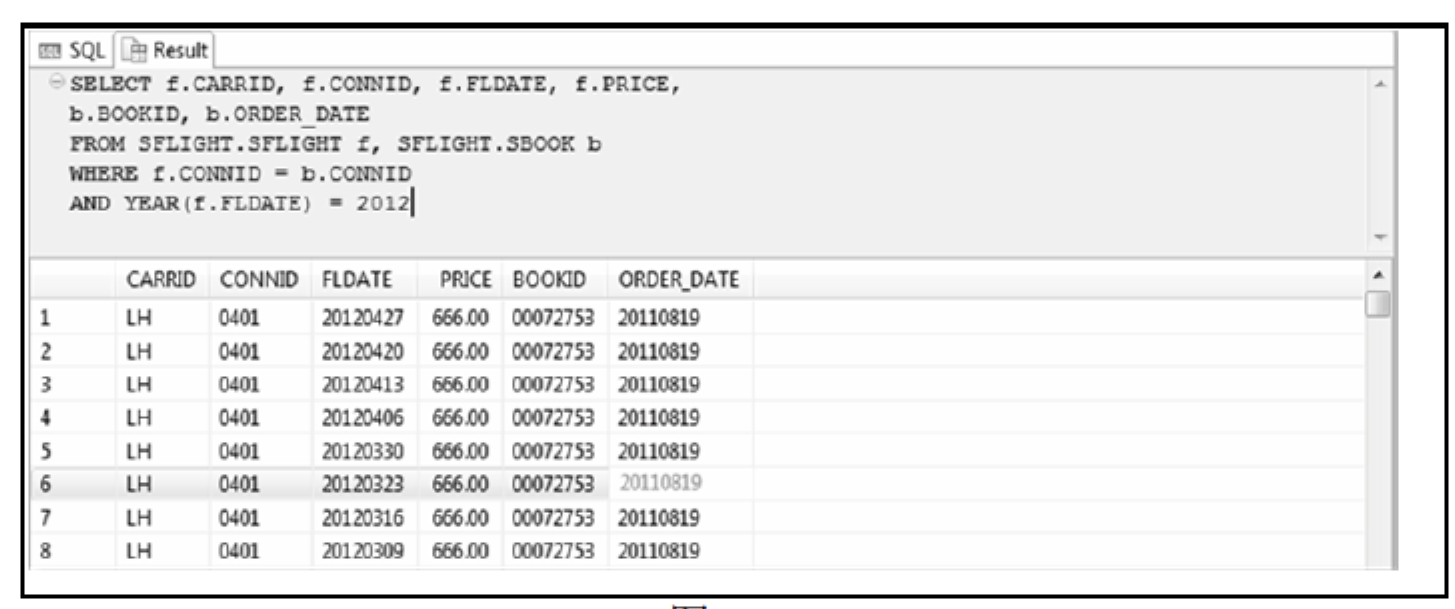


图 9-10

实例：

```
SELECT f.CARRID, f.CONNID, f.FLDATE, f.PRICE,
b.BOOKID, b.ORDER_DATE
FROM SFLIGHT.SFLIGHT f, SFLIGHT.SBOOK b
WHERE f.CONNID = b.CONNID
AND YEAR(f.FLDATE) = 2012
```

运行结果如图 9-11 所示。



SQL　Result

```
SELECT f.CARRID, f.CONNID, f.FLDATE, f.PRICE,
b.BOOKID, b.ORDER_DATE
FROM SFLIGHT.SFLIGHT f, SFLIGHT.SBOOK b
WHERE f.CONNID = b.CONNID
AND YEAR(f.FLDATE) = 2012
```

	CARRID	CONNID	FLDATE	PRICE	BOOKID	ORDER_DATE
1	LH	0401	20120427	666.00	00072753	20110819
2	LH	0401	20120420	666.00	00072753	20110819
3	LH	0401	20120413	666.00	00072753	20110819
4	LH	0401	20120406	666.00	00072753	20110819
5	LH	0401	20120330	666.00	00072753	20110819
6	LH	0401	20120323	666.00	00072753	20110819
7	LH	0401	20120316	666.00	00072753	20110819
8	LH	0401	20120309	666.00	00072753	20110819

图 9-11



外连接(Out Join)：

外连接分为左外连接、右外连接和全连接，左外连接是指结果集的数据行以左表为准，如若左表数据在右表中找不到相应的值，则返回“NULL”。右外连接与左外连接恰好相反，结果集的数据行以右表为准，如若右表数据在右左表中找不到相应的值，则返回“NULL”。全连接是指结果集将包含左右两张表中的全部数据行，缺值则以“NULL”代替，如图 9-12 所示。

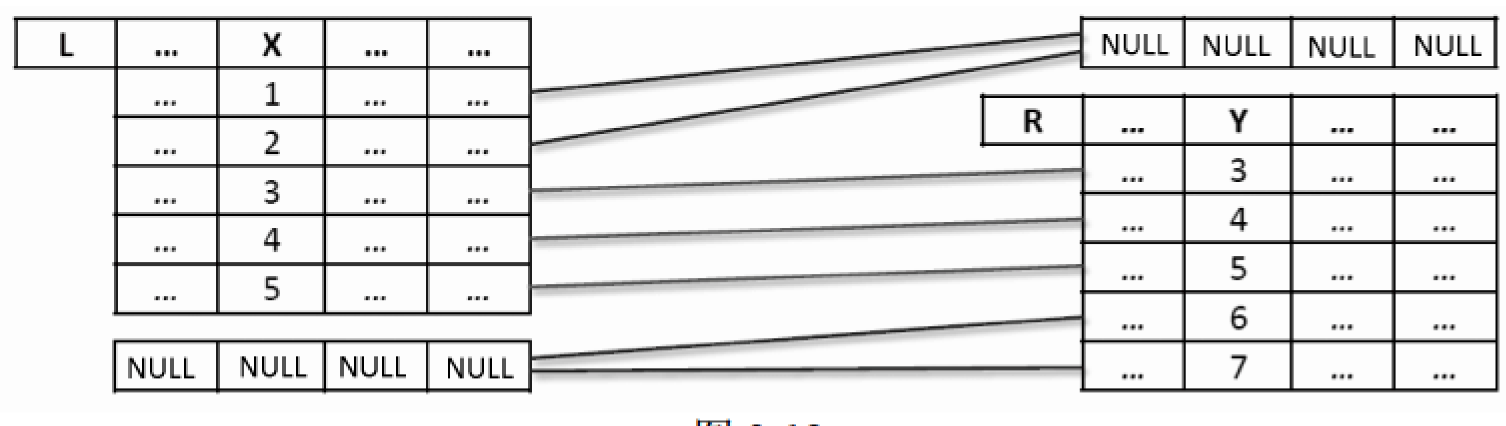
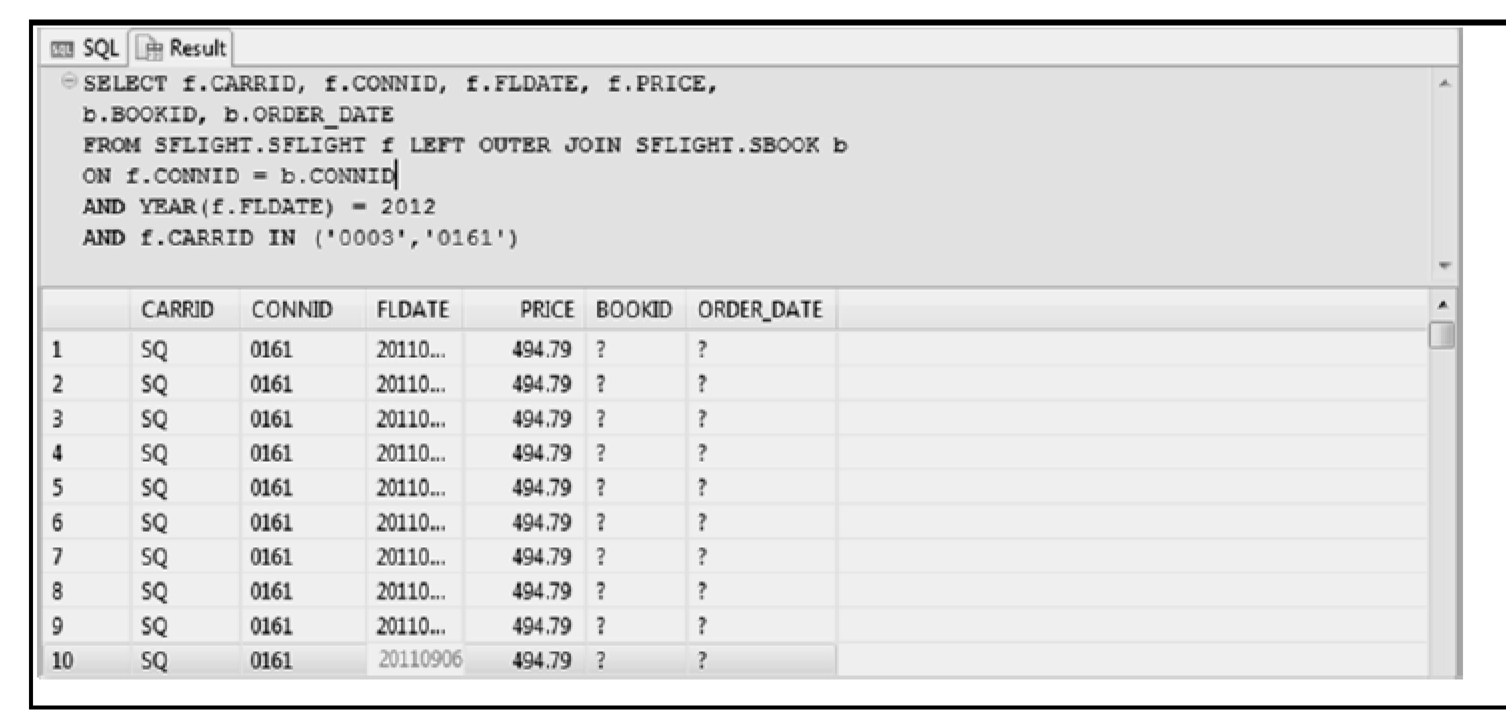


图 9-12

实例：左外连接

```
SELECT f.CARRID, f.CONNID, f.FLDATE, f.PRICE,
b.BOOKID, b.ORDER_DATE
FROM SFLIGHT.SFLIGHT f LEFT OUTER JOIN SFLIGHT.SBOOK b
ON f.CONNID = b.CONNID
AND YEAR(f.FLDATE) = 2012
AND f.CARRID IN ('0003','0161')
```

运行结果如图 9-13 所示。



SQL　Result

```
SELECT f.CARRID, f.CONNID, f.FLDATE, f.PRICE,
b.BOOKID, b.ORDER_DATE
FROM SFLIGHT.SFLIGHT f LEFT OUTER JOIN SFLIGHT.SBOOK b
ON f.CONNID = b.CONNID
AND YEAR(f.FLDATE) = 2012
AND f.CARRID IN ('0003','0161')
```

	CARRID	CONNID	FLDATE	PRICE	BOOKID	ORDER_DATE
1	SQ	0161	20110...	494.79	?	?
2	SQ	0161	20110...	494.79	?	?
3	SQ	0161	20110...	494.79	?	?
4	SQ	0161	20110...	494.79	?	?
5	SQ	0161	20110...	494.79	?	?
6	SQ	0161	20110...	494.79	?	?
7	SQ	0161	20110...	494.79	?	?
8	SQ	0161	20110...	494.79	?	?
9	SQ	0161	20110...	494.79	?	?
10	SQ	0161	20110906	494.79	?	?

图 9-13

三、复杂 SELECT 语句详解

下面我们稍微花点时间看下一些复杂的 SELECT 语句在 SAP HANA Studio 中的表现方式。图 9-14 为 SELECT 语句，后面可以使用一些 optional 参数。

```
SELECT [TOP number ] [ ALL | DISTINCT ]
        <select_list>
        <from_clause >
        [<where_clause>]
        [<group_by_clause>]
        [<having_clause>]
        [<order_by_clause>]
        [<limit_clause>]
        [<for_update_clause>]
        [<time_travel_clause>] ;
```

图 9-14

除了这些参数外，我们来通过实例了解下在 SELECT 语句中函数，CASE 语句以及聚合的使用。

- 函数(FUNCTION)

SAP HANA SQL 支持多种类别 SQL Function，常见的功能函数有时间日期函数、字符转换函数、字符串相关函数等，表 9-12 列出了几种常用的功能函数。

表 9-12

功　能	描　述
YEAR(Date)	基于提供的日期取相应的年
ADD_YEARS(Date，n)	为提供的日期加 n 年后的日期
DAYNAME(Date)	技术提供日期是星期几
CURRENT_DATE	当前日期
ABS(Number)	取绝对数
ROUND(Number)	取整
SQRT(Number)	取平方根
UPPER(String)	转换成大写
SUBSTR(String，Start，Length)	取字符串的一部分



实例：

```
SELECT YEAR (FLDATE), DAYNAME(FLDATE), CURRENT_DATE, ROUND(PRICE),
LOWER(PLANETYPE)  FROM SFLIGHT.SFLIGHT
```

运行结果如图 9-15 所示。

SQL | Result

```
SELECT YEAR (FLDATE), DAYNAME(FLDATE), CURRENT_DATE, ROUND(PRICE), LOWER(PLANETYPE)
FROM SFLIGHT.SFLIGHT
```

	YEAR(FLDATE)	DAYNAME(FLDATE)	CURRENT_DATE	ROUND(PRICE)	LOWER(PLANETYPE)
1	2011	TUESDAY	2013-05-19	495	a321
2	2011	TUESDAY	2013-05-19	495	a321
3	2011	TUESDAY	2013-05-19	495	a321
4	2011	TUESDAY	2013-05-19	495	a321
5	2011	TUESDAY	2013-05-19	495	a321

图 9-15

- CASE 语句

CASE 语句在纯 SELECT SQL 中的主要作用是基于选择列的值的不同，我们可以在输出列表中生成另外一个列。为熟悉语法，举个简单例子的如下，有点类似 ABAP 中的 hard code。

实例 1：

```
SELECT CARRID,
       CASE CARRID
           WHEN 'AC' THEN 'AIR CANANA'
           WHEN 'LH' THEN 'LUFTHANSA'
           WHEN 'BA' THEN 'BRITISH AIRWAYS'
           ELSE 'N/A'
       END AS LONGNAME,
       CONNID, FLDATE, PRICE
FROM SFLIGHT.SFLIGHT
```

运行结果如图 9-16 所示。

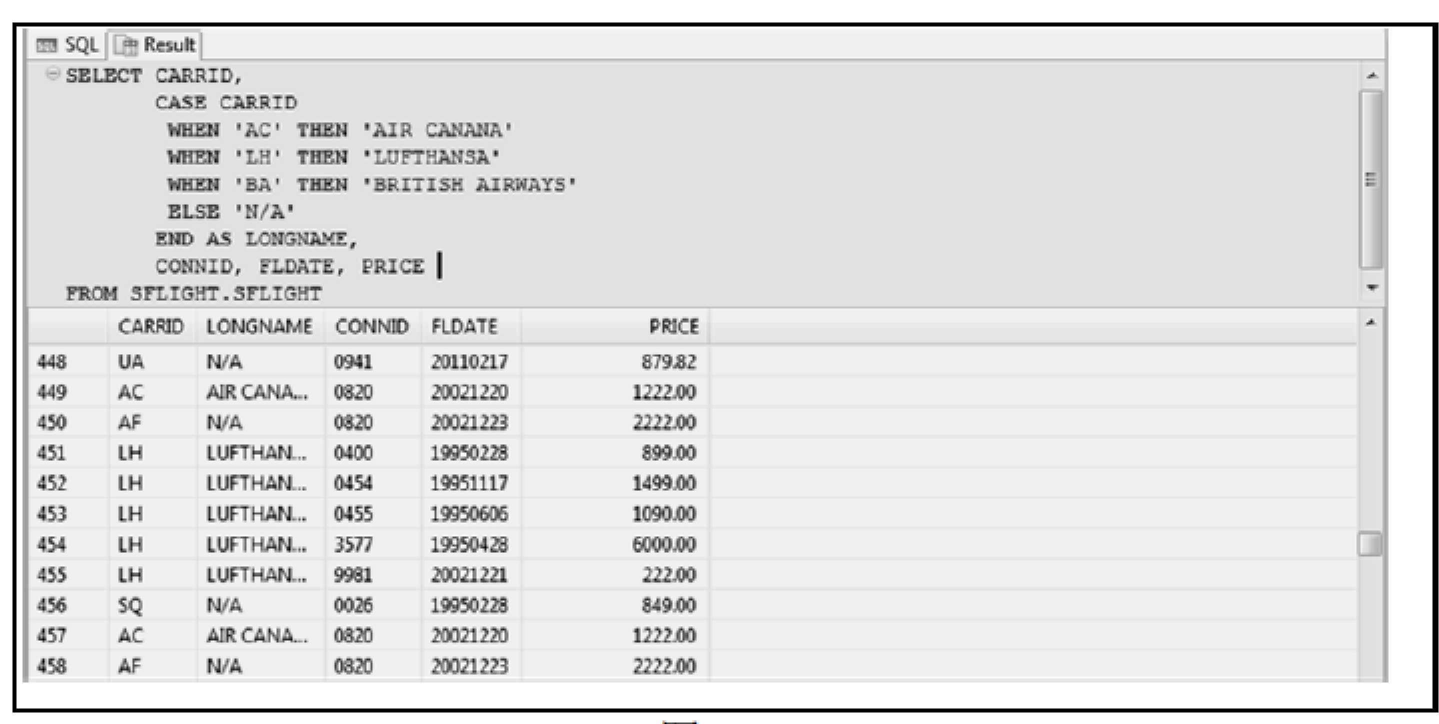


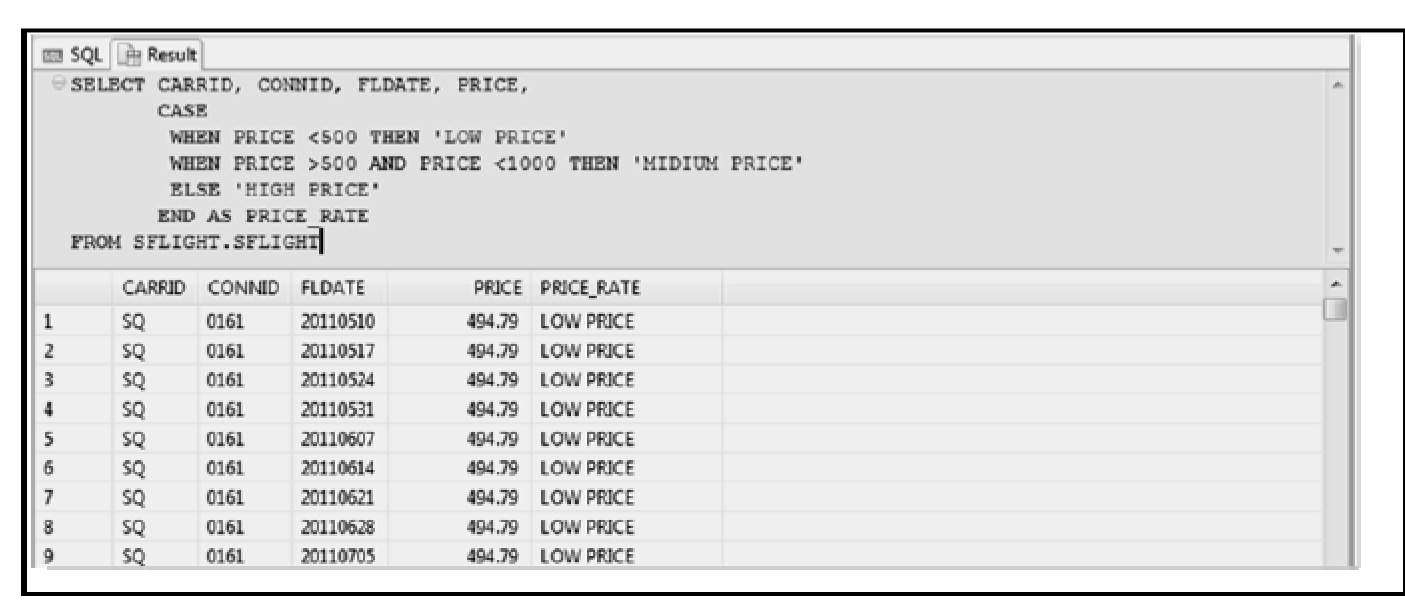
图 9-16

实例 2：一个根据某个列来分段的例子

```
SELECT CARRID, CONNID, FLDATE, PRICE,
```

```
        CASE
           WHEN PRICE <500 THEN 'LOW PRICE'
           WHEN PRICE >500 AND PRICE <1000 THEN 'MIDIUM PRICE'
           ELSE 'HIGH PRICE'
        END AS PRICE_RATE
FROM SFLIGHT.SFLIGHT
```

运行结果如图 9-17 所示。


SQL　Result

```
SELECT CARRID, CONNID, FLDATE, PRICE,
       CASE
        WHEN PRICE <500 THEN 'LOW PRICE'
        WHEN PRICE >500 AND PRICE <1000 THEN 'MIDIUM PRICE'
        ELSE 'HIGH PRICE'
       END AS PRICE_RATE
FROM SFLIGHT.SFLIGHT
```

	CARRID	CONNID	FLDATE	PRICE	PRICE_RATE
1	SQ	0161	20110510	494.79	LOW PRICE
2	SQ	0161	20110517	494.79	LOW PRICE
3	SQ	0161	20110524	494.79	LOW PRICE
4	SQ	0161	20110531	494.79	LOW PRICE
5	SQ	0161	20110607	494.79	LOW PRICE
6	SQ	0161	20110614	494.79	LOW PRICE
7	SQ	0161	20110621	494.79	LOW PRICE
8	SQ	0161	20110628	494.79	LOW PRICE
9	SQ	0161	20110705	494.79	LOW PRICE

图 9-17

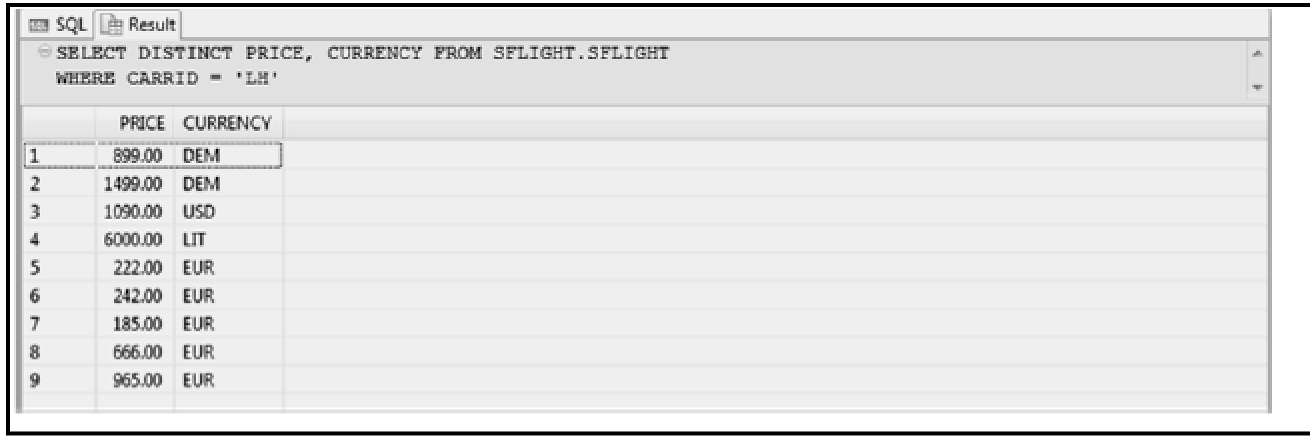
- 排除重复行项目(Duplicate Elimination)

当输出结果中不包含主键时，有可能在输出列表中会出现重复的行。我们可以使用“DISTINCT”避免重复项的出现。

实例：

```
SELECT DISTINCT PRICE, CURRENCY FROM SFLIGHT.SFLIGHT WHERE CARRID = 'LH'
```

运行结果如图 9-18 所示。


SQL　Result

```
SELECT DISTINCT PRICE, CURRENCY FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'LH'
```

	PRICE	CURRENCY
1	899.00	DEM
2	1499.00	DEM
3	1090.00	USD
4	6000.00	LIT
5	222.00	EUR
6	242.00	EUR
7	185.00	EUR
8	666.00	EUR
9	965.00	EUR

图 9-18



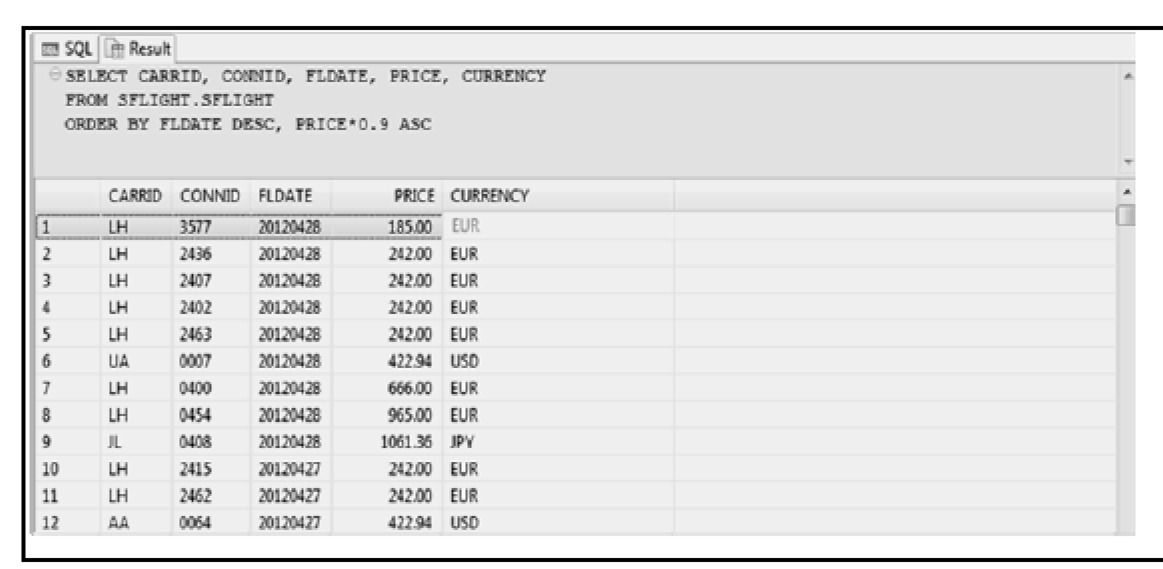
- 排序(Sorting)

在 SELECT 语句中，我们可以使用 ORDER BY DESC/ASC 对输出列表进行降序/升序排序。

实例：

```
SELECT CARRID, CONNID, FLDATE, PRICE, CURRENCY
FROM SFLIGHT.SFLIGHT
ORDER BY FLDATE DESC, PRICE*0.9 ASC
```

运行结果如图 9-19 所示。



	CARRID	CONNID	FLDATE	PRICE	CURRENCY
1	LH	3577	20120428	185.00	EUR
2	LH	2436	20120428	242.00	EUR
3	LH	2407	20120428	242.00	EUR
4	LH	2402	20120428	242.00	EUR
5	LH	2463	20120428	242.00	EUR
6	UA	0007	20120428	422.94	USD
7	LH	0400	20120428	666.00	EUR
8	LH	0454	20120428	965.00	EUR
9	JL	0408	20120428	1061.36	JPY
10	LH	2415	20120427	242.00	EUR
11	LH	2462	20120427	242.00	EUR
12	AA	0064	20120427	422.94	USD

图 9-19

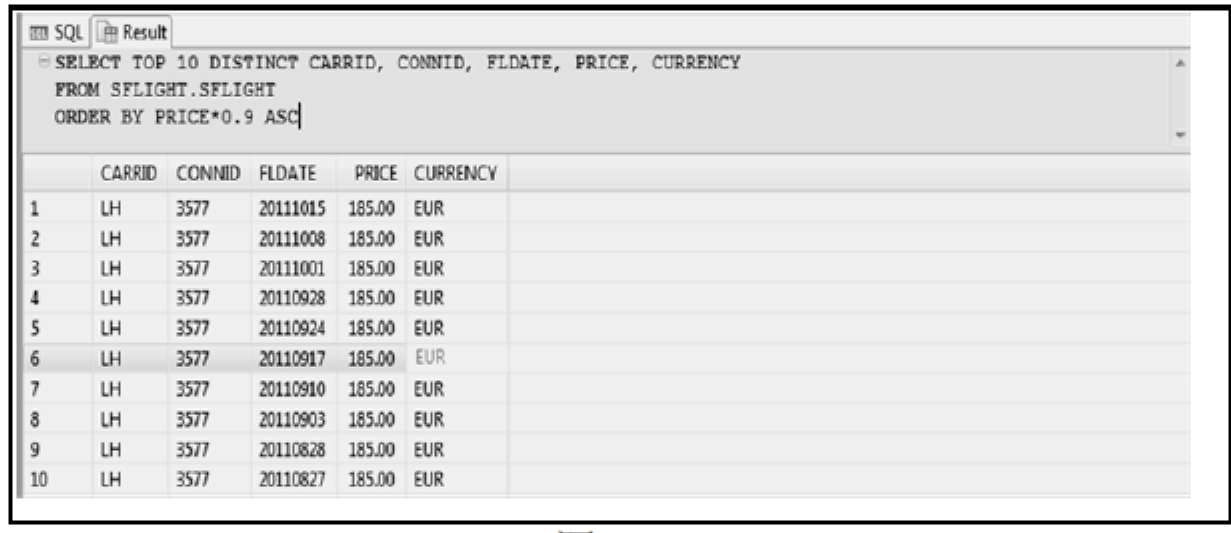
- TOP n 子句

TOP n 子句的作用是取 SELECT 语句返回结果中前 n 项。

实例：

```
SELECT TOP 10 DISTINCT CARRID, CONNID, FLDATE, PRICE, CURRENCY
FROM SFLIGHT.SFLIGHT
ORDER BY PRICE*0.9 ASC
```

运行结果如图 9-20 所示。



	CARRID	CONNID	FLDATE	PRICE	CURRENCY
1	LH	3577	20111015	185.00	EUR
2	LH	3577	20111008	185.00	EUR
3	LH	3577	20111001	185.00	EUR
4	LH	3577	20110928	185.00	EUR
5	LH	3577	20110924	185.00	EUR
6	LH	3577	20110917	185.00	EUR
7	LH	3577	20110910	185.00	EUR
8	LH	3577	20110903	185.00	EUR
9	LH	3577	20110828	185.00	EUR
10	LH	3577	20110827	185.00	EUR

图 9-20

- WHERE 从句(包括 LIKE)

我们可以在 SELECT 语句中使用 WHERE 从句来对结果集进行过滤。WHERE 后面可以使用的语句如表 9-13 所示。

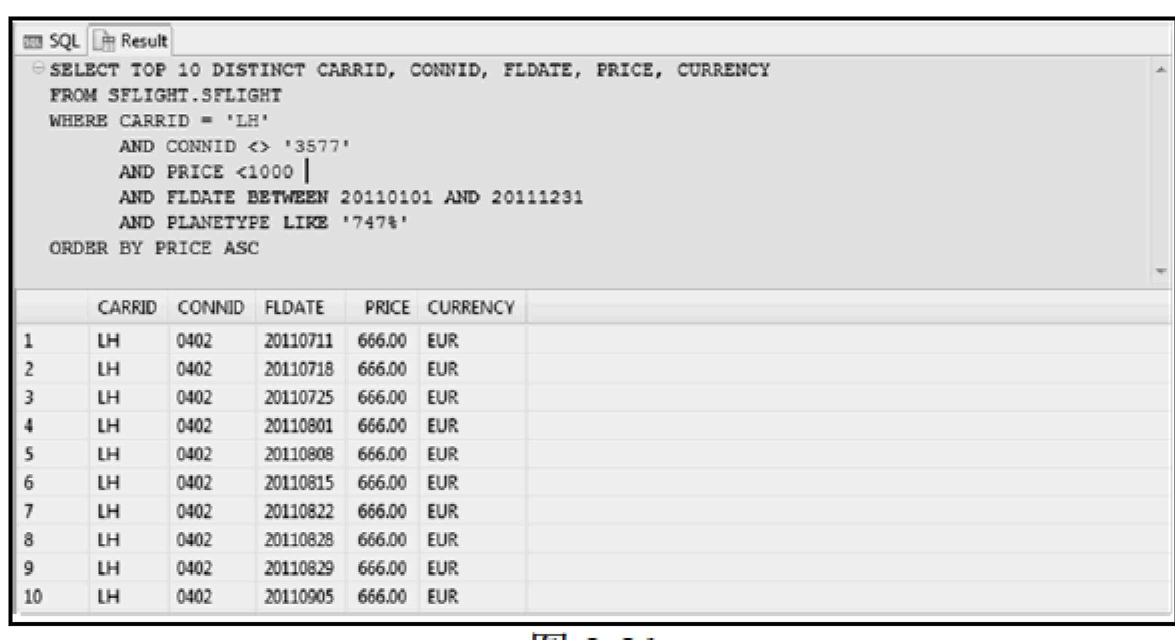
表 9-13

运 算 符	描　　述
=	相等
<> or !=	不想等
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN	在……之间
LIKE	模式匹配
IN	在某个集合内

实例：

```
SELECT TOP 10 DISTINCT CARRID, CONNID, FLDATE, PRICE, CURRENCY
FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'LH'
 AND CONNID <> '3577'
 AND PRICE <1000
 AND FLDATE BETWEEN 20110101 AND 20111231
 AND PLANETYPE LIKE '747%'
ORDER BY PRICE ASC
```

运行结果如图 9-21 所示。


SQL Result

```
SELECT TOP 10 DISTINCT CARRID, CONNID, FLDATE, PRICE, CURRENCY
FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'LH'
      AND CONNID <> '3577'
      AND PRICE <1000
      AND FLDATE BETWEEN 20110101 AND 20111231
      AND PLANETYPE LIKE '747%'
ORDER BY PRICE ASC
```

	CARRID	CONNID	FLDATE	PRICE	CURRENCY
1	LH	0402	20110711	666.00	EUR
2	LH	0402	20110718	666.00	EUR
3	LH	0402	20110725	666.00	EUR
4	LH	0402	20110801	666.00	EUR
5	LH	0402	20110808	666.00	EUR
6	LH	0402	20110815	666.00	EUR
7	LH	0402	20110822	666.00	EUR
8	LH	0402	20110828	666.00	EUR
9	LH	0402	20110829	666.00	EUR
10	LH	0402	20110905	666.00	EUR

图 9-21

这里我们对 LIKE 语句也做下简单说明：对于 LIKE 匹配，“%”代表多个字符串，“_”代表单个字符串。匹配范例如表 9-14 所示。

表 9-14

例　　子	描　　述
WHERE MyColumn LIKE 'M%'	字符串以 M 开头
WHERE MyColumn LIKE 'M _'	两位字符以 M 开头
WHERE MyColumn LIKE '%M'	字符串以 M 结尾
WHERE MyColumn LIKE '%M%'	字符串包含 M
WHERE MyColumn LIKE '_ _ _'	三位字符串
HERE MyColumn LIKE '_ _ _ _ T _ M%'	字符串第 5 位是 T，七位时 M

- 聚合表达式(Aggregate Expressions：MAX/MIN/AVG/SUM)，如表 9-15 所示。

表 9-15

聚合表达式名称	作　　用
COUNT	用来计算数据集合中的行项目 数量
MIN	取集合中最小值
MAX	取集合中最大值
SUM	取集合求和
AVG	取集合平均值
STDDEV	取集合标准差

实例：COUNT

```
SELECT COUNT(*) FROM SFLIGHT.SFLIGHT WHERE CARRID = 'SQ' AND CONNID
= '0161'
```

运行结果如图 9-22 所示。

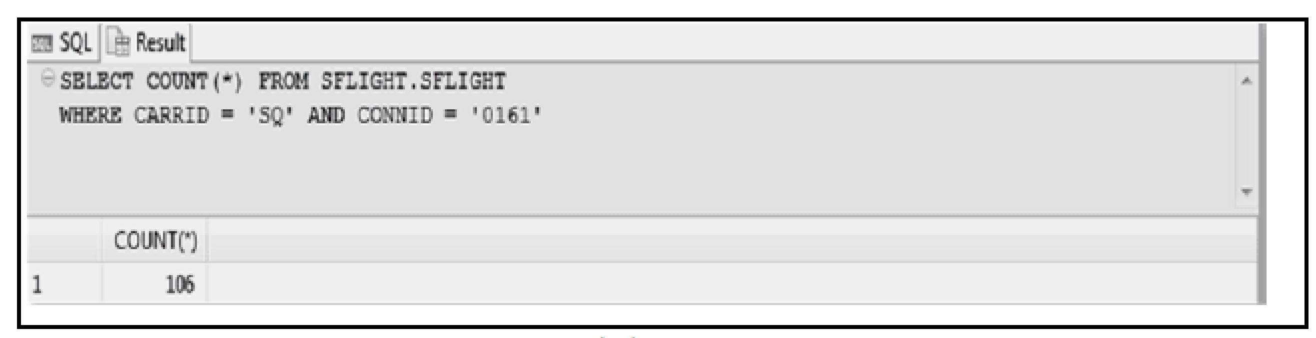


图 9-22

实例：MIN/MAX/AVG

```
SELECT MAX(PRICE), MIN(PRICE), ROUND(AVG(PRICE)) FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'SQ';
```

运行结果如图 9-23 所示。

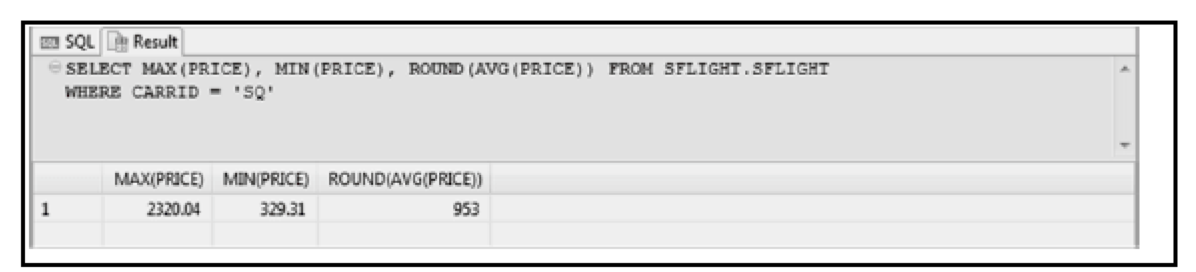


图 9-23

实例：SUM

```
SELECT SUM(SEATSMAX) FROM SFLIGHT.SFLIGHT WHERE YEAR(FLDATE)= 2012;
```

运行结果如图 9-24 所示。

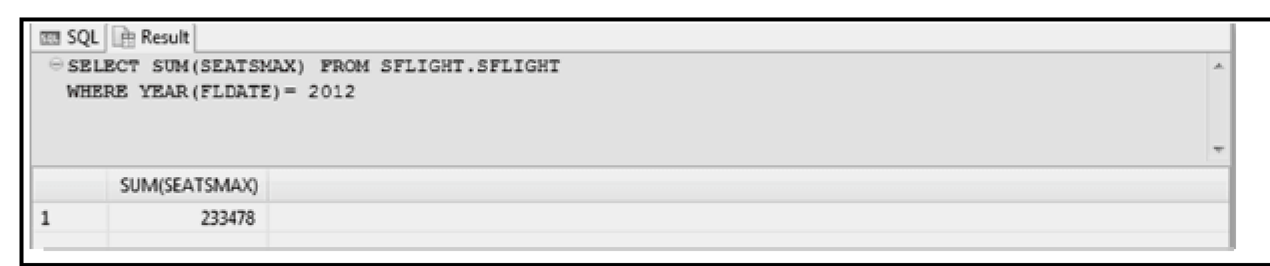


图 9-24

实例：GROUP BY

使用 GROUP BY，我们可以把数据分成不同小的数据集合，对每个小的数据集合分别做一些数据聚合(Count，Max，Min)的分析，最后用一个行项目代表每个小数据集合。

举个例子，查看各个航线中最低价格，并且要求飞机型号是 A321 或 747-400。

代码：

```
SELECT CARRID, CONNID, MIN(PRICE)
FROM SFLIGHT.SFLIGHT
WHERE PLANETYPE IN ('A321','747-400')
GROUP BY CARRID, CONNID
ORDER BY 3 ASC
```

运行结果如图 9-25 所示。

SQL　Result

```
SELECT CARRID, CONNID, MIN(PRICE)
FROM SFLIGHT.SFLIGHT
WHERE PLANETYPE IN ('A321','747-400')
GROUP BY CARRID, CONNID
ORDER BY 3 ASC
```

	CARRID	CONNID	MIN(PRICE)
1	LH	3577	185.00
2	LH	2463	242.00
3	LH	2402	242.00
4	LH	2407	242.00
5	SQ	0067	329.31
6	AA	0017	422.94
7	DL	1984	422.94
8	SQ	0161	494.79
9	SQ	0158	494.79
10	UA	3517	611.01

图 9-25



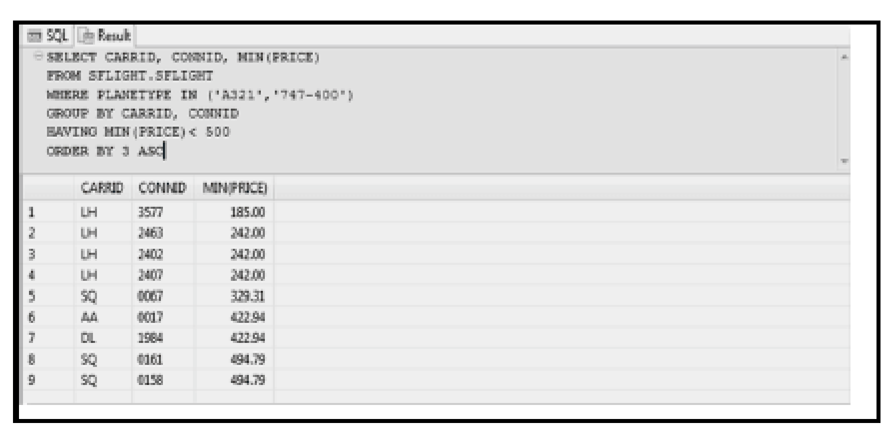
实例：HAVING

使用 HAVING，我们可以进一步对 GROUP 后的数据做筛选，比如我们只看最低价格小于 500 的航线。

代码：

```
SELECT CARRID, CONNID, MIN(PRICE)
FROM SFLIGHT.SFLIGHT
WHERE PLANETYPE IN ('A321','747-400')
GROUP BY CARRID, CONNID
HAVING MIN(PRICE)< 500
ORDER BY 3 ASC
```

运行结果如图 9-26 所示。



	CARRID	CONNID	MIN(PRICE)
1	LH	3577	185.00
2	LH	2463	242.00
3	LH	2402	242.00
4	LH	2407	242.00
5	SQ	0067	329.31
6	AA	0017	422.94
7	DL	1984	422.94
8	SQ	0161	494.79
9	SQ	0158	494.79

图 9-26

HAVING 除了可以使用 SELECT 语句引用的字段外，还可以使用 SELECT 语句中没有引用的字段。比如我们要求分组必须满足“MIN(SEATSOCC)<3”这一条件。

代码：

```
SELECT CARRID, CONNID, MIN(PRICE)
FROM SFLIGHT.SFLIGHT
WHERE PLANETYPE IN ('A321','747-400')
GROUP BY CARRID, CONNID
HAVING MIN(SEATSOCC) < 3
ORDER BY 3 ASC
```

运行结果如图 9-27 所示。

图 9-27

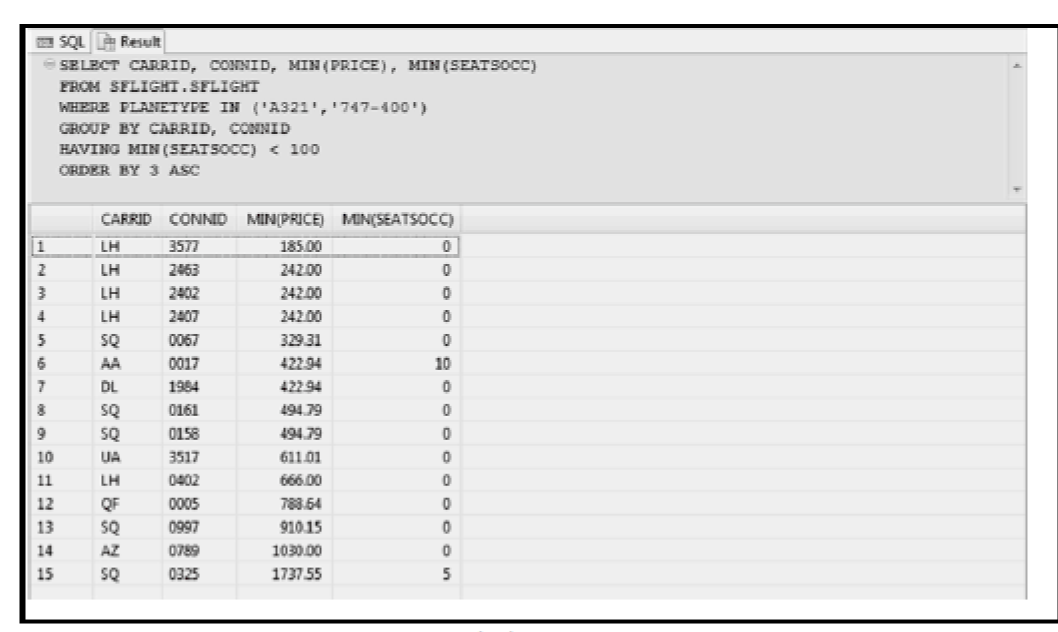


大家可以对比一下如果我们显式地把“SEATSOCC”字段包含在 SELECT 语句中，也会得到同样的效果。

代码：

```
SELECT CARRID, CONNID, MIN(PRICE), MIN(SEATSOCC)
FROM SFLIGHT.SFLIGHT
WHERE PLANETYPE IN ('A321','747-400')
GROUP BY CARRID, CONNID
HAVING MIN(SEATSOCC) < 100
ORDER BY 3 ASC
```

运行结果如图 9-28 所示。



	CARRID	CONNID	MIN(PRICE)	MIN(SEATSOCC)
1	LH	3577	185.00	0
2	LH	2463	242.00	0
3	LH	2402	242.00	0
4	LH	2407	242.00	0
5	SQ	0067	329.31	0
6	AA	0017	422.94	10
7	DL	1984	422.94	0
8	SQ	0161	494.79	0
9	SQ	0158	494.79	0
10	UA	3517	611.01	0
11	LH	0402	666.00	0
12	QF	0005	788.64	0
13	SQ	0997	910.15	0
14	AZ	0789	1030.00	0
15	SQ	0325	1737.55	5

图 9-28

四、子查询

子查询(Sub Query)可以用来辅助主查询语句来实现非常复杂的查询方式。子查询的方式可以使 SELECT 语句更加结构化和容易理解。

实例：

```
SELECT CONNID, BOOKID, ORDER_DATE FROM SFLIGHT.SBOOK
WHERE CONNID IN (SELECT DISTINCT CONNID
                 FROM SFLIGHT.SFLIGHT
                 WHERE YEAR(FLDATE)= 2012
                 AND PRICE < 300)
```

运行结果如图 9-29 所示。

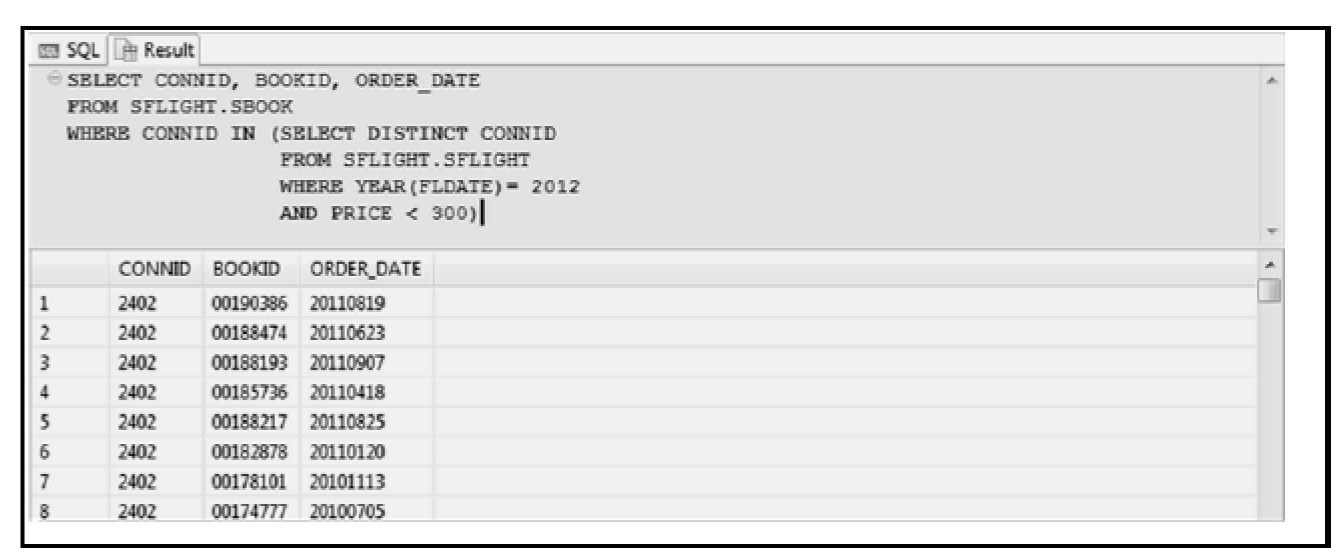


图 9-29

上面这个例子是一个简单的使用 IN 的无关联子查询(Uncorrelated Sub Query)，除了使用 IN，我们也可以使用 ANY(见表 9-16)、ALL(见表 9-17)等(IN 其实和“=ANY”逻辑上一样)。

表 9-16

= ANY	与任何一个匹配即可
< ANY	小于任何一个 (小于最大)
<= ANY	小于等于任何一个
> ANY	大于任何一个 (大于最小)
>= ANY	大于等于任何一个
<> ANY	不等于任何一个

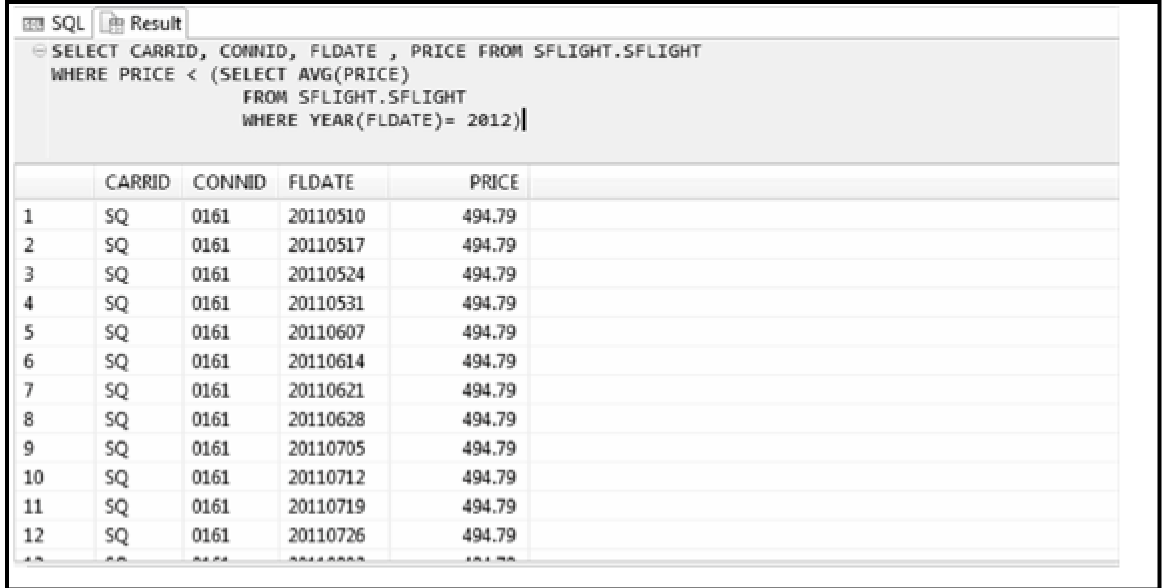
表 9-17

= ALL	与所有值匹配即可
< ALL	小于所有值 (小于最小)
<= ALL	小于等于所有值
> ALL	大于所有值 (大于最大)
>= ALL	大于所有值
<> ALL	不等于所有值

下面是一个使用“<”的无关联子查询例子。

```
SELECT CARRID, CONNID, FLDATE , PRICE FROM SFLIGHT.SFLIGHT
WHERE PRICE < (SELECT AVG(PRICE) FROM SFLIGHT.SFLIGHT WHERE
YEAR(FLDATE)= 2012)
```

运行结果如图 9-30 所示。


SQL | Result

```
SELECT CARRID, CONNID, FLDATE , PRICE FROM SFLIGHT.SFLIGHT
WHERE PRICE < (SELECT AVG(PRICE)
               FROM SFLIGHT.SFLIGHT
               WHERE YEAR(FLDATE)= 2012)
```

	CARRID	CONNID	FLDATE	PRICE
1	SQ	0161	20110510	494.79
2	SQ	0161	20110517	494.79
3	SQ	0161	20110524	494.79
4	SQ	0161	20110531	494.79
5	SQ	0161	20110607	494.79
6	SQ	0161	20110614	494.79
7	SQ	0161	20110621	494.79
8	SQ	0161	20110628	494.79
9	SQ	0161	20110705	494.79
10	SQ	0161	20110712	494.79
11	SQ	0161	20110719	494.79
12	SQ	0161	20110726	494.79

图 9-30

除了无关联子查询，我们也可以在主查询语句和子查询语句之间加上关系，也就是将它们变成关联查询。

实例 1：

```
SELECT COUNT(*) FROM SFLIGHT.SBOOK b
WHERE EXISTS (SELECT CONNID
                   FROM SFLIGHT.SFLIGHT f
                   WHERE PRICE < 200
                   AND f.CONNID = b.CONNID);
```

运行结果如图 9-31 所示。

SQL | Result

```
SELECT COUNT(*) FROM SFLIGHT.SBOOK b
WHERE EXISTS (SELECT CONNID
              FROM SFLIGHT.SFLIGHT f
              WHERE PRICE < 200
              AND f.CONNID = b.CONNID)
```

	COUNT(*)
1	60,381

图 9-31



实例 2：

```
SELECT CARRID, CONNID, FLDATE,
(SELECT MAX(PRICE) FROM SFLIGHT.SFLIGHT f1 WHERE f1.CONNID = f2.CONNID )-
PRICE AS DIFFIRENCE
FROM SFLIGHT.SFLIGHT f2
```

运行结果如图 9-32 所示。

SQL　Result

```
SELECT CARRID, CONNID, FLDATE,
(SELECT MAX(PRICE) FROM SFLIGHT.SFLIGHT f1 WHERE f1.CONNID = f2.CONNID )- PRICE AS DIFFIRENCE
FROM SFLIGHT.SFLIGHT f2
```

	CARRID	CONNID	FLDATE	DIFFIRENCE
454	LH	3577	19950428	0.00
455	LH	9981	20021221	0.00
456	SQ	0026	19950228	0.00
457	AC	0820	20021220	1000.00
458	AF	0820	20021223	0.00
459	LH	0400	19950228	0.00
460	LH	0454	19951117	0.00

图 9-32

五、更新数据

前面我们介绍了 SQL 语言关于数据表数据查询读取部分，现在简要地看一下数据更新，也就是增、删、改。

- 数据增加(INSERT)

实例：

```
INSERT INTO SFLIGHT.SCARR
VALUES ( '000', 'CA', 'China Airline', 'CNY','http://www.airchina.com');
INSERT INTO SFLIGHT.SCARR( MANDT,CARRID,CARRNAME )
VALUES ('000','PD','PUDONG' );
```

- 数据删除(DELETE)

实例：

```
DELETE FROM SFLIGHT.SCARR WHERE CARRID IN ('PV', 'CA');
```

- 数据修改(UPDATE)

实例：

```
UPDATE SFLIGHT.SCARR
SET CARRNAME = 'PUDONG_NEW'FROM WHERE CARRID = 'PD' ;
```

六、创建和修改 SQL 视图

在 DBMS 的介绍中我们提到过外部层、概念层和内部层的三层数据库管理系统架构。通过使用 SQL 视图，也就是外部层，我们可以分离终端用户和内部物理层的依赖，避免内部层的修改对用户应用的影响。同时对复杂查询也可以起到数据预处理和简化的作用。我们来看一个创建基于 SFLIGHT 表的 SQL 视图，里面只包含了价格小于 1000 元的重要航班信息，如航线、航班号、日期和价格。



实例：

```
CREATE VIEW SFLIGHT.FLIGHT AS
SELECT CARRID AS AIRLINE， CONNID AS FNUMBER， FLDATE AS FLYDATE，
PRICE，CURRENCY
FROM SFLIGHT
WHERE PRICE < 1000
WITH CHECK OPTION;
```

当这个 SQL 视图生成后，我们可以对视图像数据表一样做数据读取操作。

实例：

```
SELECT AIRLINE，FNUMBER，FLYDATE，PRICE，CURRENCY
FROM FLIGHT
WHERE AIRLINE = 'CA'
```

你同样可以对 SQL 视图做数据的增删改操作，但数据实际上是更新到相应的数据表中去的。如果是要对视图进行增删改操作，建议使用 WITH CHECK OPTION 参数以确保数据一致性。WITH CHECK OPTION 参数可以防止未满足视图条件的数据的误更新。

举两个例子：

对于插入新数据，比如我们的视图里面只包含了价格小于 1000 元的航班，没有使用 check option，我们对视图做了一个数据插入的动作，这条数据价格大于 1000，虽然对视图的操作会显示成功，数据也会被更新到相应的数据表，但这个视图又看不到这条更新的数据。



对于 DELETE 操作，我们删除一条价格大于 1000 的数据，虽然对视图的操作会显示成功，但实际上这条数据并没有被从表中删除。

当我们不需要某个视图时，删除此视图即可。

实例：

```
DROP VIEW SFLIGHT.FLIGHT;
```

第四节　数据控制语言

前面大家可能注意到当管理员新建一个用户时，系统会自动创建和用户名一样的 Schema，这个 Schema 是此用户的缺省 Schema。我们一般不会明显注明是从哪个 Schema

读取数据，但是我们也可以注明，像前面提到的例子 SFLIGHT.FLIGHT。我们也可以从对多个 Schema 的数据进行操作。比如我们想把一个 Schema 下某个表示的数据复制到另一个 Schema 的表下面，可以这样做：

```
INSERT INTO TEST.SFLIGHT ( MANDT, CARRID, CONNID, FLDATE, PRICE, CURRENCY )
SELECT MANDT, CARRID, CONNID, FLDATE, PRICE, CURRENCY FROM SFLIGHT.SFLIGHT
WHERE CARRID = 'LH';
```

像 ABAP 程序一样，有时我们需要控制数据的访问权限，定义那些用户可以看到哪些数据。对 SAP HANA 而言我们有两种方式，一种就是创建一个新的视图，把这个用户需要看到的数据放入这个视图中，另一种就是为这个用户创建相应的对表的读取权限。SAP HANA SQL 中主要的权限控制语句是 GRANT 和 REVOKE。

- GRANT

通过 GRANT 我们可以授予用户对某个表和字段的读取权限。例子如下(WITH GRANT OPTION 指该被授权用户可以将此授权再授权给其他用户)。

实例：

```
GRANT SELECT, INSERT, UPDATE ON TEST.SFLIGHT
TO JEVINSKI
WITH GRANT OPTION;

GRANT SELECT (CARRID, CONNID) ON TEST.SFLIGHT
TO JEVINSKI
WITH GRANT OPTION;
```

对不同粒度大小的对象，我们可以授予的权限如表 9-18 所示。

表 9-18

粒　度	权　限
(single) Schema	SELECT , INSERT , UPDATE , DELETE , DROP , ALTER , CREATE ANY
(single) Table	SELECT , INSERT , UPDATE , DELETE , DROP , ALTER
(single) View	SELECT , INSERT , UPDATE , DELETE , DROP
(single) Column	SELECT , INSERT , UPDATE

- REVOKE

REVOKE 的语法和 GRANT 类似。

实例：

```
REVOKE SELECT, INSERT, UPDATE ON TEST.SFLIGHT
FROM JEVINSKI;
```



第五节　关于 NULL 值

“NULL”是 SQL 中一个特殊的预留字。NULL 不等于数字的 0 或者字符串的空格。NULL 值会在如下几种情况下产生：

- INSERT /UPDATE 数据表时，某列的值没有提供，或者明确指出该列值为 NULL。
- OUTER JOIN

对于某数值和 NULL 值的比较操作，我们需要格外小心。对比如 NULL AND 关系操作的例子(见表 9-19)。

表 9-19

X	Y	X AND Y
true	true	true
true	false	false
True	unknown	unknown
false	true	false
false	false	false
false	unknown	false
unknown	true	unknown
unknown	false	false
unknown	unknown	unknown

再举个具体的例子，比如有个表“employee”，其中 Overtime 字段记录了员工加班的时间，如下两个查询的值是不同的。前者 Overtime 包含 NULL，而后者没有。

```
SELECT Name, Overtime
FROM employee;

SELECT Name, Overtime
FROM employee
WHERE Overtime <= 10 OR Overtime > 10;
```



本章小结与练习

在 SQL 简介部分，我们了解了什么是 SQL、DBMS 和 DBMS 的三层架构。然

后以最简单的问题“为什么关系型数据库要叫做库‘关系’型数据库”开始了解什么是关系(relation)，提醒开发人员需要将以前过程语言编程中对单一数据操作的编程思维习惯转换到以数据集合(关系为操作对象)的 SQL 编程思维方式。最简单的例子是直接使用 SUM 或 AVG 求和或取平均，而不必循环对数据一一操作。

接下来我们从 DDL、DML 和 DCL 三个方面对 SQL 做了相应的阐述。

首先，在 DDL 定义数据部分，我们了解了用来定义数据的基本数据类型：

- 时间日期(DATE，TIME，TIMESTAMP)
- 数字(INTEGER/DECIMAL)
- 字符串(VARCHAR，NVARCHAR，SHORTTEXT)
- 二进制和大数据对象类型(BLOB，CLOB，TEXT)

之后我们了解了如何用这些数据类型来创建表，并简要地看了如何修改表，以及如何删除和重命名表。

然后，在 DML 操作数据部分，我们从简单的单表 SELECT 说起，再到稍复杂的 SELECT 展开，比如：

- SQL Function(数据类型转换，日期，数字，和字符串相关 Function)
- CASE ...WHEN ...THEN... END AS ... 语句
- DISTINCT
- ORDER BY ...ASC/DESC
- TOP N
- WHERE(=，<>，BETWEEN，IN，LIKE(通配符% _))
- 聚合(COUNT/MIN/MAX/SUM/AVG/STDDEV，GROUP BY/HAVING)

关于多表的 UION 和 JOIN 操作，我们简要了解了更新数据部分(INSERT/UPDATE/DELETE)。除了基于表的操作外，我们还学习了视图的创建以及 WITH CHECK OPTION 对视图数据更新的用途。

最后，在 DCL 访问控制部分，简要地说明了 GRANT 和 REVOKE 的用法。

除了上述 DDL/DML/DCL 主要内容，NULL VALUE 也是 SQL 中的一个重要概念，我们也简要地了解了 NULL 的特殊性和其注意事项。

练习

1. 先为练习尝试创建一个新节点，名称可以自己定义。
2. 将新的节点设为当前缺省的节点，并预览查看确认。
3. 参考 EPM 中的如下几项，通过“create table”在新的节点中创建你自己的表。



purchase order header/item

product

businessPartner

4. 对单表 product item 做一些 SELECT 查询，尝试 WHERE 和 Aggregation 操作，比如某个产品在某个时间段内的“price”总和、最大值、最小值等。

5. 以 productID 字段 Join PO 表和 product 表，或以 supplierID 字段 Join product 表和 businessPartner 表，选择你感兴趣的字段比如价格做一些 Aggregation 操作，比如供应商 A 的产品 B 和 C 在某年的所有 PO 的价格总和。

6. 将连接的方式换成 SubQuery 的方式重新操作一次。

7. 尝试对上面连接后显示的字段新建一个视图，并将此视图的读取授予某个用户。



第十章　SAP HANA SQL Script

第一节　SQL Script 简介

大家可能知道，SAP HANA 的创新除了融合内存计算、列存储、查询优化以及并行运算，还有一个重要的基本理念就是将大数据的处理和计算逻辑从应用层搬到基于 Memory 的数据库层(如图 10-1 所示)，仅将需要在 UI 上看到的数据，如计算结果传递到应用层，从而避免大量数据在数据库层和应用层的往返复制和传输。

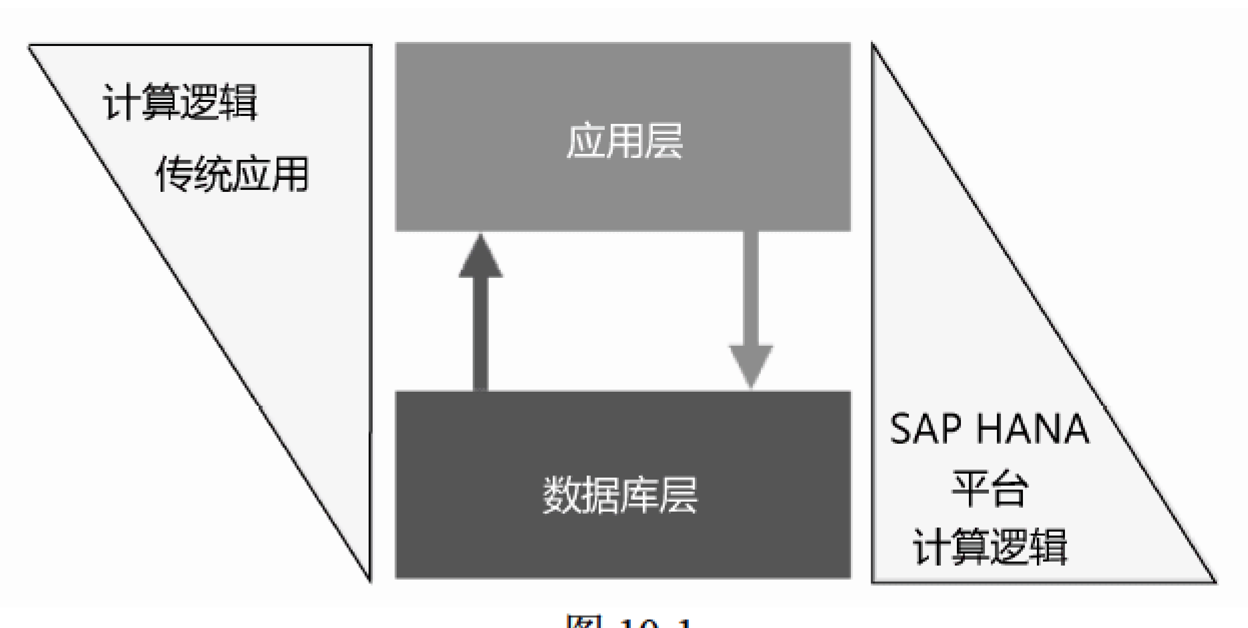


图 10-1

传统的应用程序一般都是将业务逻辑(比如计算)，放在应用层，数据库层仅用来存储数据。我们熟知的传统的 SAP ERP、ABAP 应用程序、基于 Java 的应用程序都是如此。对大量数据，传统应用程序先把这些数据从数据库层转移到应用层的一个大的内表后，再做计算和处理。如 Loop 循环，每次处理单条数据，然后再把结果返回到数据库层或在 UI 上展示。我们可以称这种方式为数据到代码(Data to Code，如图 10-2 左所示)，也就是先把数据在计算执行所在的层(应用层)准备好，再做业务逻辑计算(Code)。而 SAP HANA 新的方式可以称之为代码到数据(Code to Data，如图 10-2 右所示)，也就是直接在数据所在的层(In Memory DB 层)做大数据



业务逻辑计算，仅将需要的少量数据传送给 UI 层，应用层业务逻辑这个层次被极大的弱化，因为大数据的计算被下放(push down)到了数据库层。图 10-2 是从传统数据到代码方式转换到代码到数据方式的图形化描述，我们可以看到代码所在层次从应用层到数据库层的转变。

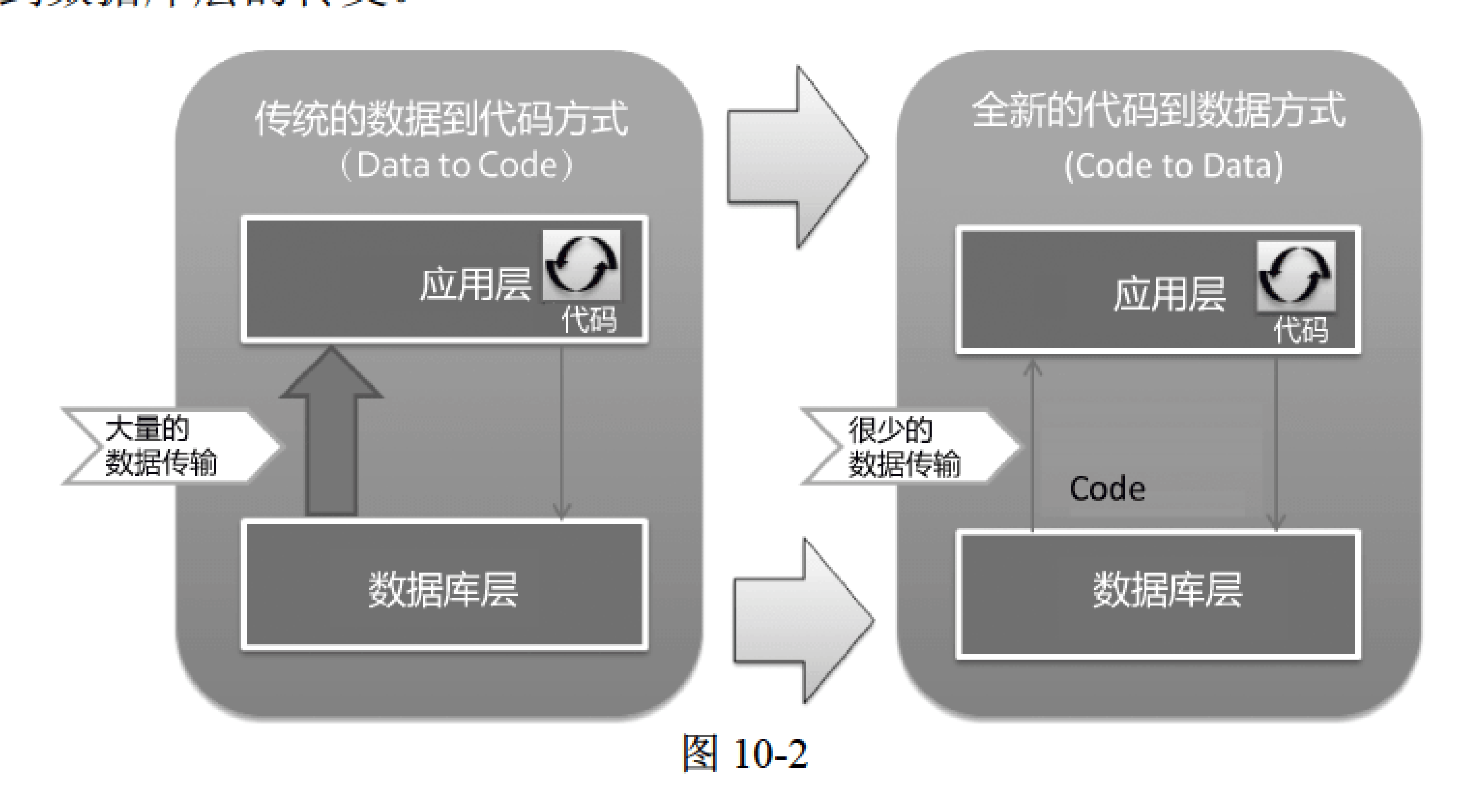


图 10-2

而这个创新想法和转变需要技术实现基础。开发人员需要使用某种语言或工具在数据库实现复杂计算逻辑(比如数据库存储过程)，并支持像“IF ELSE”这样的控制结构语句。为了实现这个在数据库层进行复杂计算逻辑的目标，SAP 对标准的 SQL 做了如下三个方面的扩展，扩展后的语言也就是 SAP HANA SQL Script，在本书中我们简称其为“SQL Script”。使用 SQL Script，开发人员可以开发在 SAP HANA 中可重用的存储过程，即 SAP HANA 存储过程。



SQL Script 对标准 SQL 的扩展具体包括：

- 数据类型扩展：容许定义“Table Type”数据类型，即使还没有相应的数据库基础表。
- 函数扩展：容许处理复杂数据流，比如支持多个输入输出，存储过程嵌套。
- 规则扩展：容许规则程序的执行，逻辑控制结构语句和数据更新。

对于开发人员而言，相对 SQL，最直接的感受时就是 SQL Script 有如下扩展和优点：

- 一个 SQL 查询仅能返回一个数据集，而存储过程可以返回多个数据集。
- 对于复杂的查询，SQL 仅能使用 SQL 视图来构造实现，但视图没有相应输入输出参数可以使用；而使用 SQL Script，我们可以定义多个输入输出，进

行模块化，拆分复杂的函数逻辑，将复杂逻辑细分成子逻辑，增强程序的结构性、可读性和重用性。

- SQL Script 支持对中间步骤数据定义临时本地变量，而 SQL 需要定义全局视图，即使是为中间临时数据使用。
- SQL Script 支持规则控制，比如 IF/ELSE、LOOP 等。

为了更好地理解 SQL Script 和存储过程，我们先看下存储过程是如何在 SAP HANA 系统里编译处理和执行的。以创建存储过程语句为例，SAP HANA 系统有这样一些处理步骤：

- 解析语句，检测和报告简单语法错误。
- 检查语句的语义正确性、变量类型和其使用的一致性。
- 代码优化：HANA 内置优化引擎会区分上游的声明逻辑(Declarative logic)和下游业务流程逻辑(Orchestration logic)。声明逻辑指 SELECT 查询和内置的计算引擎函数。业务流程逻辑指 DDL(数据定义语言)、DML(数据操纵语言)、变量的分配和其他命令逻辑，比如 IF ELSE、LOOP 等。后面我们会简要介绍 SAP HANA 是如何区分处理声明逻辑和业务流程逻辑的。
- 代码生成：对于声明逻辑编译器会创建相应的计算模式(calculation models)和数据流图(Data flow graph)。这些计算模式会被计算引擎进一步优化。
- 前一步生成的声明式计算模式会在系统中累积生成堆栈。
- 编译后生成的存储过程实时对象会存放在 SAP HANA 的目录和资源库中。

而调用执行存储过程的时候也有两个步骤：

- 编译：当存储过程被调用的时候，存储过程会再次被计算引擎编译。
- 执行：而当执行开始时，实际的参数会被传递到前面定义阶段生成的计算模型中，当计算模型被实例化时，基于参数提供的实际输入值，计算引擎会做进一步优化。

图 10-3 说明了 SAP HANA 调用和执行存储过程的整体过程，即从编译开始，到生成数据流图，到在计算引擎中做优化和执行。图中提及的 SAP HANA 计算引擎是 SQL Script 的执行引擎。



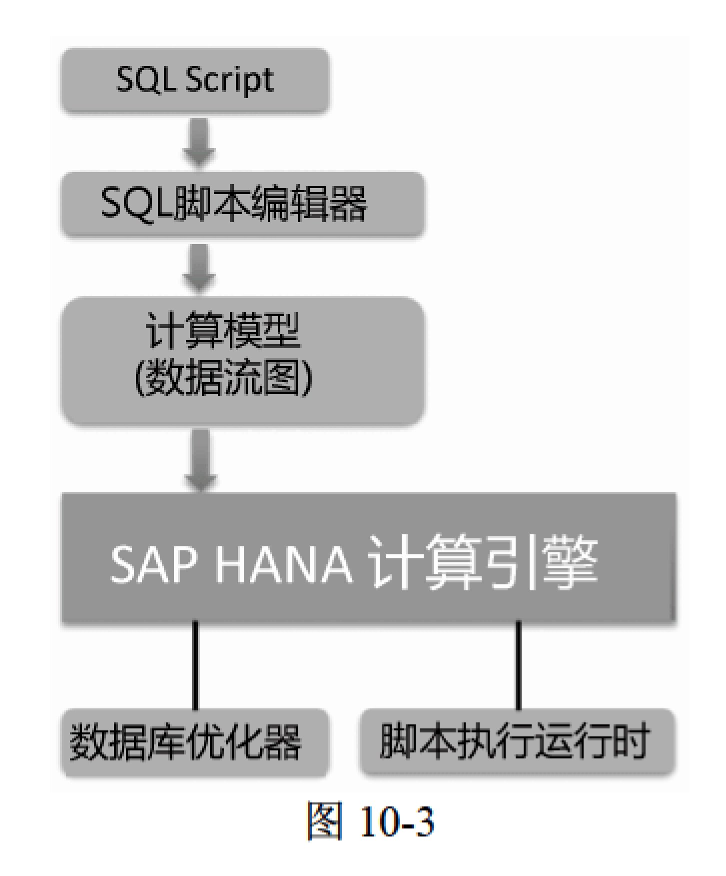


图 10-3

图 10-4 是计算模型的一个例子。如图所示，计算模型是一个单方向的、无循环的数据流图，包含数据输入、数据操作节点，如 Join/Aggregation/ Projection/Union 等。单方向无循环的数据流图也意味着循环和递归在计算模型中不容许一个节点的输出是另一个节点的输入，节点之间没有双向数据传递。除了 Join/Aggregation/Projection 这些数据操作节点外，计算模型可以包含其他类型节点，比如：

- SQL 节点，用于执行某些 SQL 语句。
- R 节点，用于处理数据统计相关。
- Scripting 节点，对于某些特别复杂和特殊的计算逻辑，可以用其他语言实现如 Python、JavaScript 等。



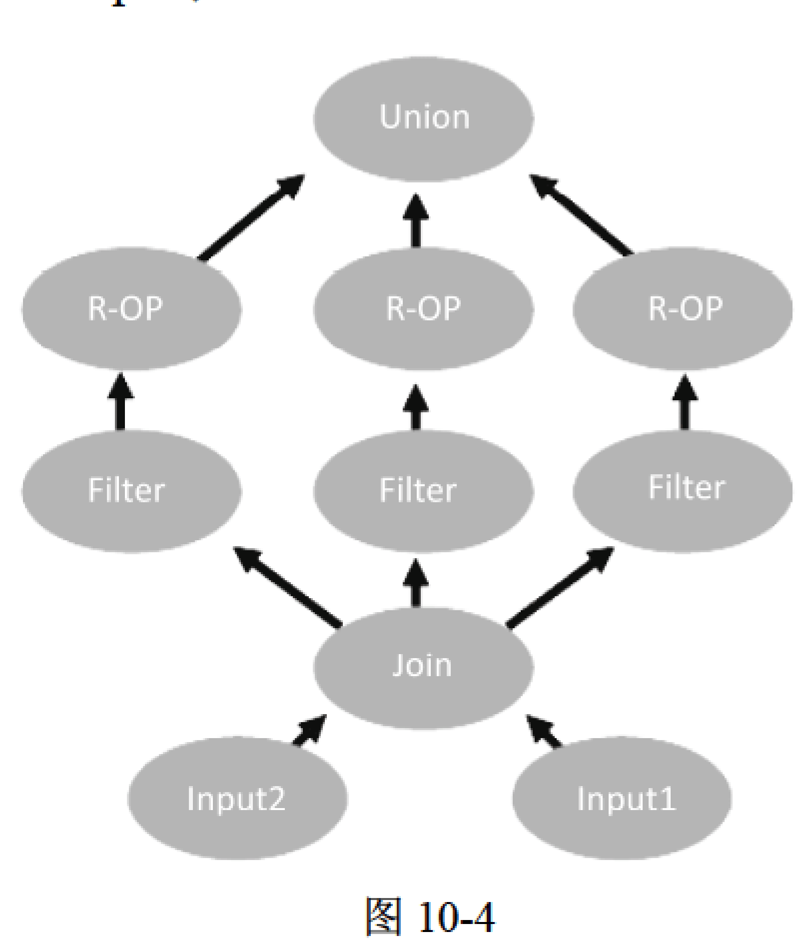


图 10-4

上面提到的内容大家只需大概了解，接下来我们来详细介绍下业务流程逻辑和声明逻辑的不同。对于用于执行目的声明类逻辑，SAP HANA 数据库系统会优化后再执行，而业务流程逻辑则是用于协调目的的命令类逻辑，SAP HANA 内存数据库系统会按照顺序依次执行，优化空间较少。下面我们稍微展开了解下业务流程逻辑和声明逻辑。

一、业务流程逻辑和声明逻辑

业务流程逻辑和声明逻辑如图 10-5 所示。

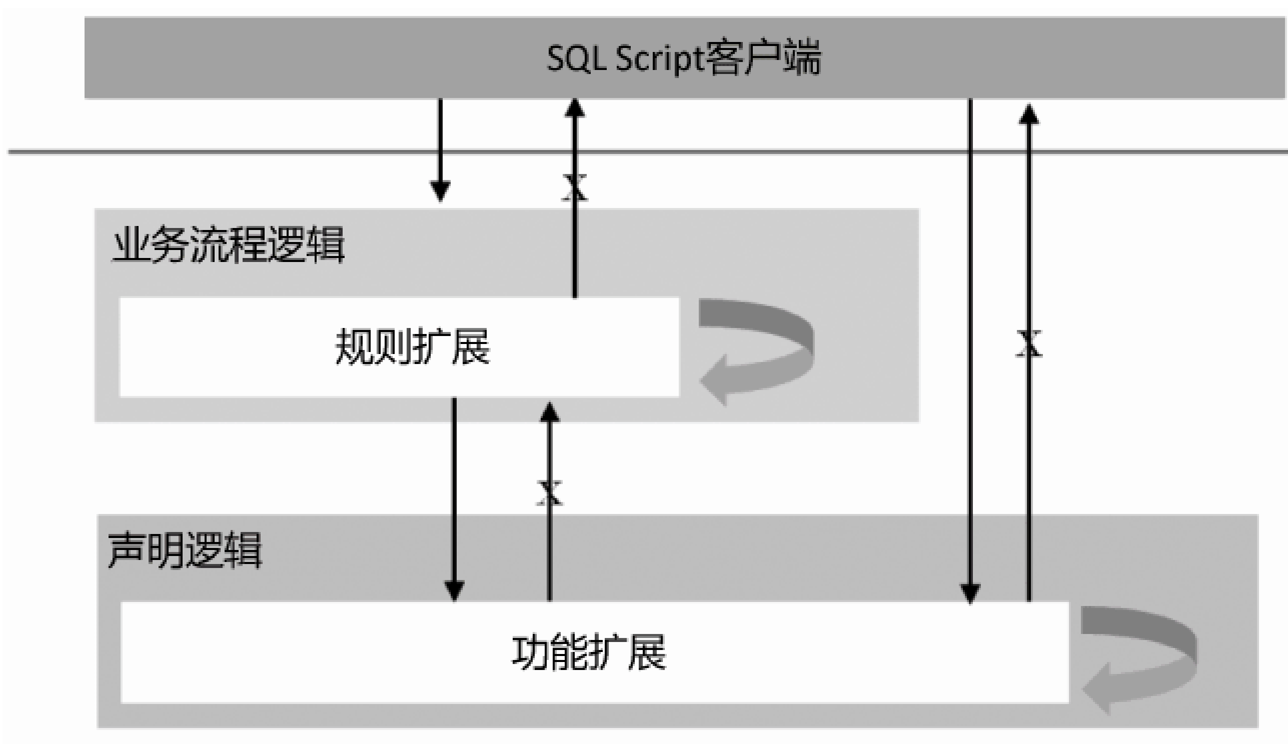


图 10-5

业务流程逻辑用于协调目的的命令类逻辑，是指通过 DDL、DML、查询语句和规则语言结构(比如 LOOP 和条件语句)来实现和控制数据流。

为了达到更好的执行效率，所有的 SAP HANA SQL Script 语句会在最大程度上被转变成 SAP HANA 易于理解的数据流图。而编译时，查询 SELECT 相关的声明逻辑代码片段会被从中抽离出来，计算引擎会执行此部分计算，并且该计算可以优化和并行。

如上所述，SAP HANA 把计算逻辑转换成数据流图，并做相应的优化。但优化有两个前提：

- 所有数据流操作必须是边际效应无关的，也就是说在计算逻辑中不能有对全局数据的更新操作。
- 所有的数据控制流必须可以转换成静态的数据流图，不能有动态的计算逻辑。



声明逻辑用于执行目的声明类逻辑。相对于业务流程逻辑的协调和组织特征，声明逻辑则负责高效执行大数据计算。在数据流图中声明逻辑是可以被并行运算的。并行运算的前提也类似，相关的操作必须是边际效应无关的，这也意味着在整个程序中不能有更新全局数据的逻辑存在。基于这样的前提，如下的一些运算式我们称之为声明逻辑：

- SQL SELECT 语句
- CE Function 语句
- SAP 提供的一些客户定制化语句

了解了 SAP HANA 的业务流程逻辑和声明逻辑，以及相应的 SAP HANA 优化前提，现在我们开始进一步了解 SQL Script 是如何对 SQL 进行扩展的。

二、SAP HANA SQL Script 对标准 SQL 的扩展

我们先看看在数据类型方面的扩展，SQL Script 除了支持前面提到如下标准 SQL(92)数据类型，如：

- <TINYINT，SMALLINT，INTEGER，BIGINT>
- <DECIMAL(p，s)，REAL，FLOAT，DOUBLE>
- <VARCHAR，NVARCHAR，CLOB，NCLOB>
- <VARBINARY，BLOB>
- <DATE，TIME，TIMESTAMP>

还支持 Table Type 类型的数据。Table Type 类型可用在存储过程的输入和输出。使用“create type”我们可以定义需要的数据集合类型。



Table Type 实例：

```
CREATE TYPE tt_publishers AS TABLE (
   publisher INTEGER,
   name VARCHAR(50),
   price DECIMAL,
   cnt INTEGER);
```

1. 函数扩展(Functional Extension)

简单来说，函数方面的扩展是指 SAP HANA SQL Script 容许开发人员编写类似函数的重用类程式，比如类似 SAP ABAP 的事务代码“SE37”所写出的函数模型，可以自定义多个输入、输出参数，并且输入输出也可以是前面提及的表类型的数据

结构。

一般而言，从概念上说，函数方面扩展的存储过程仅指的是只读数据相关的数据读取和转换，没有数据更新，并且不包含规则命令类语句，比如 IF ELSE、WHILE 循环等。

2. 规则命令类语句扩展(Imperative Extension)

如果我们不仅仅需要对数据集合做一些转换，还需要对全局数据做一些更新，甚至需要声明，使用本地的标量变量和一些规则命令类语言，比如 IF ELSE 这样的结构控制语句，从概念上说，这就不仅是函数扩展，而是规则命令类语句扩展。

了解了 SQL Script 对标准 SQL 三个方面扩展的基本概念后，我们开始正式进入 SAP HANA SQL Script 和存储过程部分。

第二节　SAP HANA 存储过程

存储过程(Procedure)可以很简单，对 ABAP 开发了解的读者的可以把存储过程看成一个用 SQL Script 写的读取数据的 ABAP 函数模块，给一些输入条件，系统输出合乎条件的数据(当然根据情况也可以包含数据更新)。

就拿大家都熟悉的 SFLIGHT 为例，我们输入航班号和其他条件，系统输出一些 BOOKING 数据。

- 输入参数：航班年份，价格
- 输出参数：已定航班列表 TT_BOOKINGS，字段包括包含预定编号(BOOKID)，预订日期(ORDER_DATE)，旅客姓名(PASSNAME)，航班公司(CARRID)，航班(CONNID)，航班日期(FLDATE)

我们需要先为输出的数据定义表结构，代码如下：

```
CREATE TYPE SFLIGHT.TT_BOOKINGS AS TABLE (
       "CARRID" NVARCHAR (3),
       "CONNID" NVARCHAR (4),
       "FLDATE" NVARCHAR (8),
       "BOOKID" NVARCHAR (8),
       "CLASS" NVARCHAR (1),
       "ORDER_DATE" NVARCHAR (8),
       "PASSNAME" NVARCHAR (25) )
```

接下来创建存储过程主体，代码如下：

```
CREATE PROCEDURE GET_BOOKINGS( IN FLIGHT_PRICE DECIMAL(15,2),
                              IN FLIGHT_YEAR VARCHAR(3),
                              OUT OUT_BOOKINGS TT_BOOKINGS )
LANGUAGE SQLSCRIPT READS SQL DATA AS
BEGIN

SELECT CARRID, CONNID, FLDATE, BOOKID, CLASS, ORDER_DATE, PASSNAME
FROM SFLIGHT.SBOOK
WHERE CONNID IN (SELECT DISTINCT CONNID
                        FROM SFLIGHT.SFLIGHT
                        WHERE YEAR(FLDATE)= :FLIGHT_YEAR
                        AND PRICE < :FLIGHT_PRICE)

END;
```

上面的例子仅仅含有一句 SELECT 语句，提供一些直接的感性认识。下面我们开始介绍创建存储过程的细节。在开始创建存储过程前，我们先来看看编写存储过程相关的语法和创建方式。

一、存储过程语句和创建方式

编写存储过程的相关语法有：

- 创建语句 CREATE

实例：

```
CREATE PROCEDURE <proc_name> [(<parameter_clause>)]
[LANGUAGE <lang>] [SQL SECURITY <mode>] [READS SQL DATA [WITH
RESULT VIEW <view_name>]] AS <local_scalar_variables>
BEGIN
<Procedure_code>
END
```



我们将这个语句拆分开解释：

- CREATE PROCEDURE <proc_name> [(<parameter_clause>)]

上面例子已经介绍，给出存储过程名称和输入输出参数及其数据类型。

- [LANGUAGE <lang>]

指定存储过程用什么语言，一般为 SQL Script，也可以是其他语言，比如 R 语言。

- [SQL SECURITY <mode>]

指定存储过程执行的安全模式，可以是以存储过程 DEFINER 定义者的权限去执行，也可以设定为以调用者 INVOKER 权限去执行。

➢ [READS SQL DATA]

指定该存储过程为只读存储过程，没有更新数据的相关的 DML 语句或者 DDL 语句。

➢ [WITH RESULT VIEW <view_name>]

用来将读取的数据结果写入一个新的视图中，后面会详细介绍。

➢ AS <local_scalar_variables>

可以用来定义本地 local 变量

- 删除语句 DROP

实例：

```
DROP PROCEDURE <proc_name> [<drop_option>]
```

- 更新语句 ALTER

实例：

```
ALTER PROCEDURE <proc_name> RECOMPILE [WITH PLAN]
```

上面给的例子是在 SQL Console 中使用 SQL 来创建存储过程，还有另外一种方式就是使用图形化工具来创建存储过程，具体步骤为：

(1) 在 SAP HANA Studio 的“SAP HANA Systems”视图中，展开“Content”节点，在选中的包目录下，右击选择“New”→“Procedure”，如图 10-6 所示。

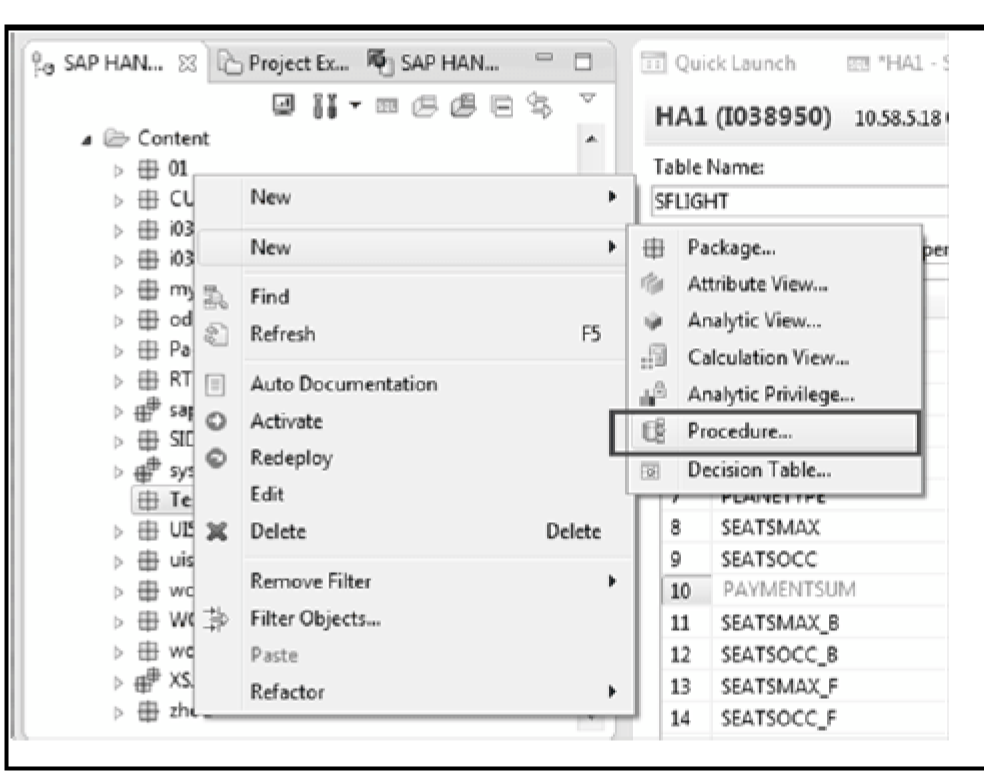
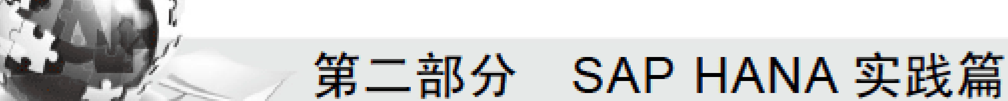


图 10-6

(2) 然后在“Script View”视图中为存储过程定义逻辑，并在旁边的面板中定义

输入输出(见图 10-7)。

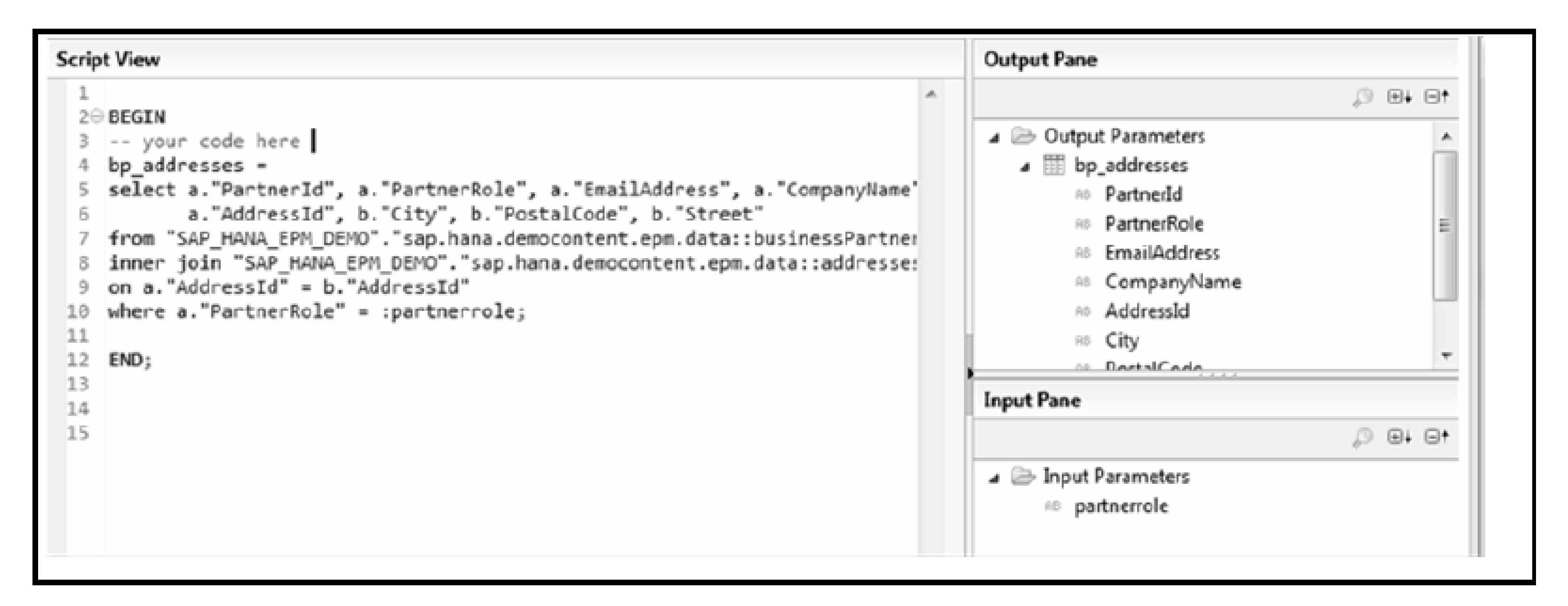


图 10-7

从 SAP HANA Studio SPS05 起，我们还可以在本地项目中通过创建后缀为“.procedure”文件的方式创建存储过程，下图是已经创建好的一个在“Project Explore”下的存储过程(见图 10-8)。



图 10-8

当我们创建了存储过程的文件后，系统会自动跳到如图 10-9 所示的类似模板的一个页面。

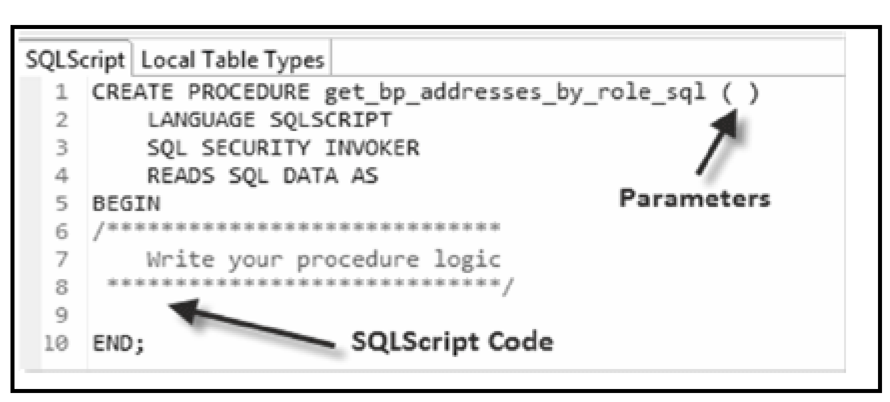


图 10-9



可以看到，除了存储过程主体外(SQL Script 分页)，SPS05 还引入了给此存储过程使用的本地数据表类型(“Local Table Types”分页)，编辑界面如图 10-10 所示。

```
SQLScript | Local Table Types
/*
You can create new table types and use them as input and output parameters in a procedure.
The table types must be defined using SQL syntax, for example:
CREATE TYPE <Type Name> AS TABLE (<Field Name1> <Field Type1>, <Field Name2> <Field Type2>,...);
*/

CREATE TYPE TT_BOOKINGS AS TABLE (
        "CARRID" NVARCHAR (3),
        "CONNID" NVARCHAR (4),
        "FLDATE" NVARCHAR (8),
        "BOOKID" NVARCHAR (8),
        "CLASS" NVARCHAR (1),
        "ORDER_DATE" NVARCHAR (8),
        "PASSNAME" NVARCHAR (25) );
```

图 10-10

使用定义好的本地数据表类型作为输出，实现具体 SQL Script 逻辑(见图 10-11)。

```
SQLScript | Local Table Types
CREATE PROCEDURE test (IN FLIGHT_PRICE DECIMAL(15,2),
                       IN FLIGHT_YEAR VARCHAR(3),
                       OUT OUT_BOOKINGS TT_BOOKINGS )
    LANGUAGE SQLSCRIPT
    SQL SECURITY INVOKER
    READS SQL DATA AS
BEGIN
/*****************************
    Write your procedure logic
 *****************************/
SELECT CARRID, CONNID, FLDATE, BOOKID, CLASS, ORDER_DATE, PASSNAME
FROM SFLIGHT.SBOOK
WHERE CONNID IN (SELECT DISTINCT CONNID
                 FROM SFLIGHT.SFLIGHT
                 WHERE YEAR(FLDATE)= :FLIGHT_YEAR
                 AND PRICE < :FLIGHT_PRICE);

END;
```

图 10-11

虽然编辑界面和方式略有不同，但从 SQL Script 语法和逻辑实现来说，没有什么变化。本章后面给出的例子和代码片段基本上都是在 SQL Console 中直接通过“create procedure” 语法的方式实现。

二、案例介绍：出版商和书籍

为了方便后面 SQL Script 语法举例，我们使用 SAP 官方 SQL Script 指南中的案例——出版商和书籍用例来作为本章的使用案例，并对官方未提及的基本信息，如提到的基本表和其字段的说明给予补充。在本章结尾，我们同样会做一个知识点小结和使用 EPM 来做练习巩固。现在我们先解释下官方 SQL Script 帮助文档中的基本表。

在本案例中，主要的表有两张：“PUBLISHERS”和“BOOKS”(出版商和书籍)如图 10-12 所示。

- “PUBLISHERS”表主要字段：出版商 ID，出版商名称，街道，邮编，城市，国家。界面如图 10-12 所示。

Table Name: PUBLISHERS　Schema: SQLSCRIPTEXAMPLE　Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null	Default	Comment
1	PUB_ID	INTEGER		INT	X(1)	X		
2	NAME	VARCHAR	50	STRING				
3	STREET	VARCHAR	50	STRING				
4	POST_CODE	VARCHAR	10	STRING				
5	CITY	VARCHAR	50	STRING				
6	COUNTRY	VARCHAR	50	STRING				

图 10-12

- “BOOKS”表主要字段：ISBN，书名，出版商，版本，发行年份，价格，货币。界面如图 10-13 所示。

Table Name: BOOKS　Schema: SQLSCRIPTEXAMPLE　Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null	Default	Comment
1	ISBN	VARCHAR	20	STRING	X(1)	X		
2	TITLE	VARCHAR	50	STRING				
3	PUBLISHER	INTEGER		INT				
4	EDITION	INTEGER		INT				
5	YEAR	VARCHAR	4	STRING				
6	PRICE	DECIMAL		FIXED				
7	CRCY	VARCHAR	3	STRING				

图 10-13



还有两张表是有声书和折扣书，这里先不展开。我们先看看如何创建这两张主表。

三、创建主表

1. PUBLISHERS(出版商)

- 先创建 SCHEMA，创建 Schema 的代码如下：

```
DROP SCHEMA SqlScriptExample CASCADE;
CREATE SCHEMA SqlScriptExample;
```

创建完成后，可以执行 SELECT 语句来检测创建是否成功：

```
SELECT CURRENT_SCHEMA "current schema" FROM DUMMY;
```

运行结果如图 10-14 所示。

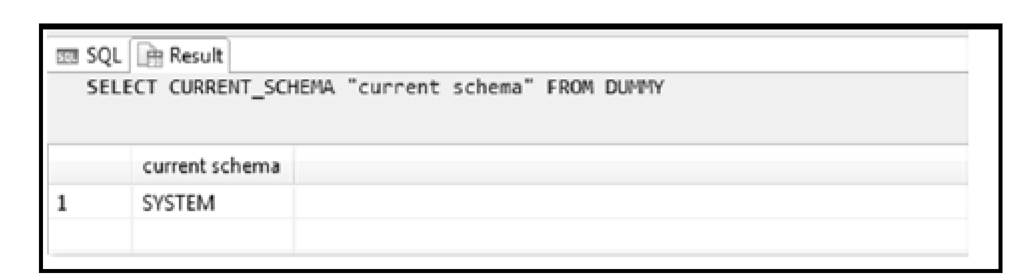


图 10-14

把新建的 SCHEMA——“SqlScriptExample”设为当前默认的 Schema，这样在后面操作我们就不需要显式的写明 SCHEMA 的名称。具体代码如下：

```
SET SCHEMA SqlScriptExample;
```

运行结果如图 10-15 所示。

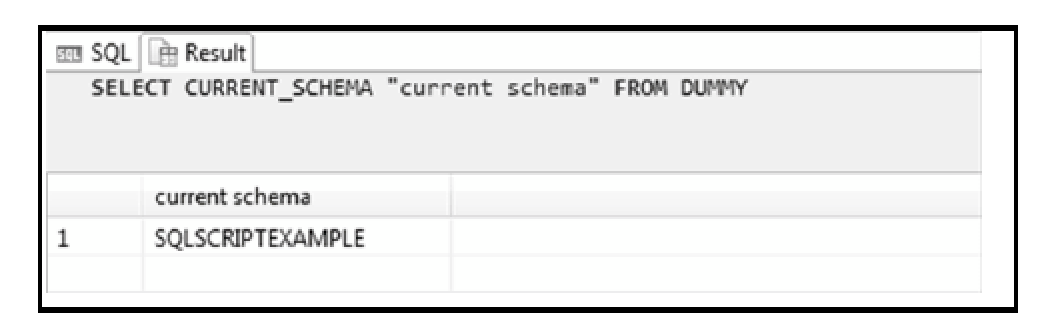


图 10-15

- 创建主表“PUBLISHERS”：

```
CREATE COLUMN TABLE publishers(
    pub_id INTEGER PRIMARY KEY,
    name VARCHAR(50),
    street VARCHAR(50),
    post_code VARCHAR(10),
    city VARCHAR(50),
    country VARCHAR(50));
```

填充数据：

```
INSERT INTO publishers VALUES
( 1, 'SAP Press', 'Chenhui Road 1001', '201203', 'Shannghai',
'China');
INSERT INTO publishers VALUES
( 2, 'Tsinghua Uni Press', 'Tshinghua Road 1001', '100084',
'beijing', 'China');
INSERT INTO publishers VALUES
( 3, 'BeiJing Uni Press', 'Peking Road 1001', '100084', 'beijing',
'China');
INSERT INTO publishers VALUES
```

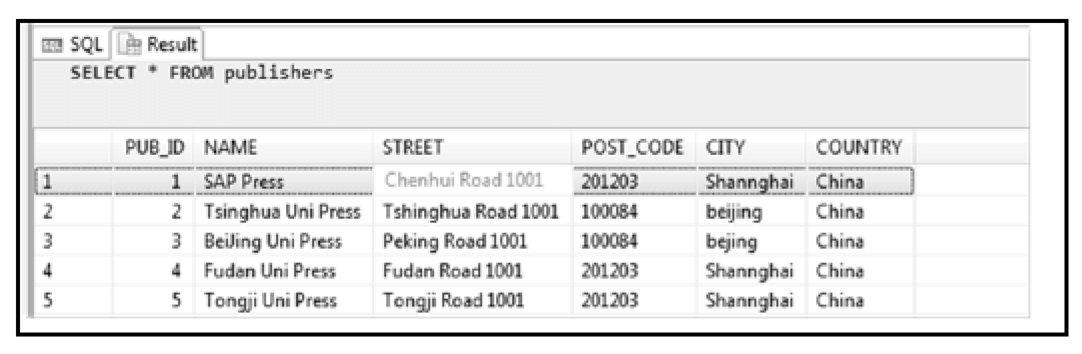
```
    ( 4, 'Fudan Uni Press', 'Fudan Road 1001', '201203', 'Shannghai',
'China');
    INSERT INTO publishers VALUES
    ( 5, 'Tongji Uni Press', 'Tongji Road 1001', '201203', 'Shannghai',
'China');
```

上面我们填充了一些练习数据，你也可以根据需要创建更多，或者通过 CSV 方式批量导入。在实际的项目中，数据一般是通过 SLT 或数据服务组件的 ETL Tool 批量导入，这里仅为练习讲解方便，我们直接通过 INSERT 来填充数据。

检查插入的数据：

```
SELECT * FROM publishers;
```

最终结果如图 10-16 所示。


SQL Result
SELECT * FROM publishers

	PUB_ID	NAME	STREET	POST_CODE	CITY	COUNTRY
1	1	SAP Press	Chenhui Road 1001	201203	Shannghai	China
2	2	Tsinghua Uni Press	Tshinghua Road 1001	100084	beijing	China
3	3	Beiling Uni Press	Peking Road 1001	100084	bejing	China
4	4	Fudan Uni Press	Fudan Road 1001	201203	Shannghai	China
5	5	Tongji Uni Press	Tongji Road 1001	201203	Shannghai	China

图 10-16

2. BOOKS(书籍)

- 创建主表“BOOKS”：

```
CREATE COLUMN TABLE books(
    isbn VARCHAR(20) PRIMARY KEY,
    title VARCHAR(50),
    publisher INTEGER,
    edition INTEGER,
    year VARCHAR(4),
    price DECIMAL(5, 2),
    crcy VARCHAR(3));
```

填充数据：

```
    INSERT INTO books VALUES ('978-3-486-57690-0', 'HANA APP DEV GUIDE',
1, 6, '2006', '39.80', 'RMB');
    INSERT INTO books VALUES ('978-3-86894-012-1', 'IN MEMORY DATABASE',
2, 3, '2009', '29.95', 'RMB');
    INSERT INTO books VALUES ('978-3-8266-1664-8', 'HANA ADMIN GUIDE',
3, 3, '2008', '39.95', 'RMB');
```



```
INSERT INTO books VALUES ('978-3-8266-1665-8', 'Mobility Dev', 1, 3, '2008', '49.95', 'RMB');
INSERT INTO books VALUES ('978-3-8266-1665-9', 'iOS Dev', 4, 3, '2010', '42.95', 'RMB');
INSERT INTO books VALUES ('978-3-8266-1665-2', 'Andriod Dev', 5, 3, '2011', '22.95', 'RMB');
INSERT INTO books VALUES ('978-3-8266-1665-4', 'SAP SUP Intro', 1, 3, '2012', '24.55', 'RMB');
```

检查插入的数据(见图 10-17):

```
SELECT * FROM books;
```

ISBN	TITLE	PUBLISHER	EDITION	YEAR	PRICE	CRCY
978-3-486-...	HANA APP DEV GUIDE	1	6	2006	39.8	RMB
978-3-8689...	IN MEMORY DATABASE	2	3	2009	29.95	RMB
978-3-8266...	HANA ADMIN GUIDE	3	3	2008	39.95	RMB
978-3-8266...	Mobility Dev	1	3	2008	49.95	RMB
978-3-8266...	iOS Dev	4	3	2010	42.95	RMB
978-3-8266...	Andriod Dev	5	3	2011	22.95	RMB
978-3-8266...	SAP SUP Intro	1	3	2012	24.55	RMB

图 10-17

3. AUDIOBOOKS(有声读物)

- 创建主表“AUDIOBOOKS”：

```
CREATE COLUMN TABLE audiobooks(
    isbn VARCHAR(20) PRIMARY KEY,
    title VARCHAR(50),
    publisher INTEGER,
    year VARCHAR(4),
    price DECIMAL(5, 2),
    crcy VARCHAR(3));
```

填充数据:

```
INSERT INTO audiobooks VALUES ('978-39388781-37-1', 'Time management', 4, '2006', '24.90', 'EUR');
```

检查插入的数据(见图 10-18):

```
SELECT * FROM audiobooks;
```



	ISBN	TITLE	PUBLISHER	YEAR	PRICE	CRCY
1	978-39388781-37-1	Time management	4	2006	24.9	EUR

图 10-18



四、创建表类型

数据准备完毕，我们再来看看存储过程的输入输出。

业务需求：假设现在需要计算和查看主流出版商(发行数量超过一定指标，数量用户指定)发行的书籍和价格总和。

- 输入：已发行书的数量，价格货币
- 输出：①出版商发行书籍的名称和其价格②某年发行书籍的名称和其价格

从输出需求中我们可以看到存储过程输出会使用到 TABLE TYPE，我们事先创建准备好。针对输出①，我们定义一个表格结构“tt_publishers”，输出②定义表格结构“tt_years”。

定义的语句如下：

```
CREATE TYPE tt_publishers AS TABLE (
   publisher INTEGER,
   name VARCHAR(50),
   price DECIMAL,
   cnt INTEGER);
CREATE TYPE tt_years AS TABLE (
   year VARCHAR(4),
   price DECIMAL,
   cnt INTEGER);
```

除了创建表结构外，我们还需要准备消息类型作为 Log 文件的基础：

```
CREATE TABLE message_box(  message  VARCHAR(200), log_time   TIMESTAMP);
```



初始化消息区的存储过程：

```
CREATE PROCEDURE init_proc LANGUAGE SQLSCRIPT AS
BEGIN
DELETE FROM message_box;
END;
```

增加消息的存储过程：

```
  CREATE  PROCEDURE  ins_msg_proc  (p_msg  VARCHAR(200))  LANGUAGE
SQLSCRIPT AS
  BEGIN
 INSERT INTO message_box VALUES (:p_msg, CURRENT_TIMESTAMP);
  END;
```

五、第一个存储过程的例子

现在准备就绪，我们来看如下存储过程的例子的主体，看看如何读取数据为输出做准备。本例中的业务需求是查看主流出版商(发行数量超过一定指标，数量用户指定)发行的书籍和价格总和。

存储过程代码：

```
CREATE PROCEDURE getOutput( IN cnt INTEGER,
                            IN currency VARCHAR(3),
                            OUT output_pubs tt_publishers,
                            OUT output_year tt_years)
LANGUAGE SQLSCRIPT SQL SECURITY DEFINER READS SQL DATA AS
BEGIN

  --Query1 取得满足条件的出版商 ID 列表(以"--"开头为注释行)

  big_pub_ids = SELECT publisher AS pid
                FROM books
                GROUP BY publisher
                HAVING COUNT(isbn) > :cnt;

  --Query2 取得根据 ID 列表和其他输入值读取书籍相关数据

  big_pub_books = SELECT title, name, publisher, year, price
                  FROM :big_pub_ids, publishers, books
                  WHERE pub_id = pid
                  AND pub_id = publisher
                  AND crcy = :currency;

   --Query3 以出版商为基础对 Query 2 的数据做聚合 → 价格 SUM

  output_pubs = SELECT publisher, name, SUM(price) AS price,
COUNT(title) AS cnt
                FROM :big_pub_books
                GROUP BY publisher, name;
   --Query4 以时间年份为基础，对 Query 2 数据做聚合 → 价格 SUM

  output_year = SELECT year, SUM(price) AS price, COUNT(title) AS
cnt
                FROM :big_pub_books
                GROUP BY year;
END;
```

从这个例子可以看到，基于 SQL Script 的存储过程可以定义多个输入，多个输

出。输入一般为标量类型数据，也可以包含表类型，输出为表类型。对查询主体而言，前面查询生成的结果可以赋值给临时表变量，继续被后面的查询直接使用。例子中的查询 1 的结果 big_pub_ids 可以给查询 2 使用，查询 2 的结果 big_pub_books 可以给查询 3 和查询 4 使用，如图 10-19 所示。

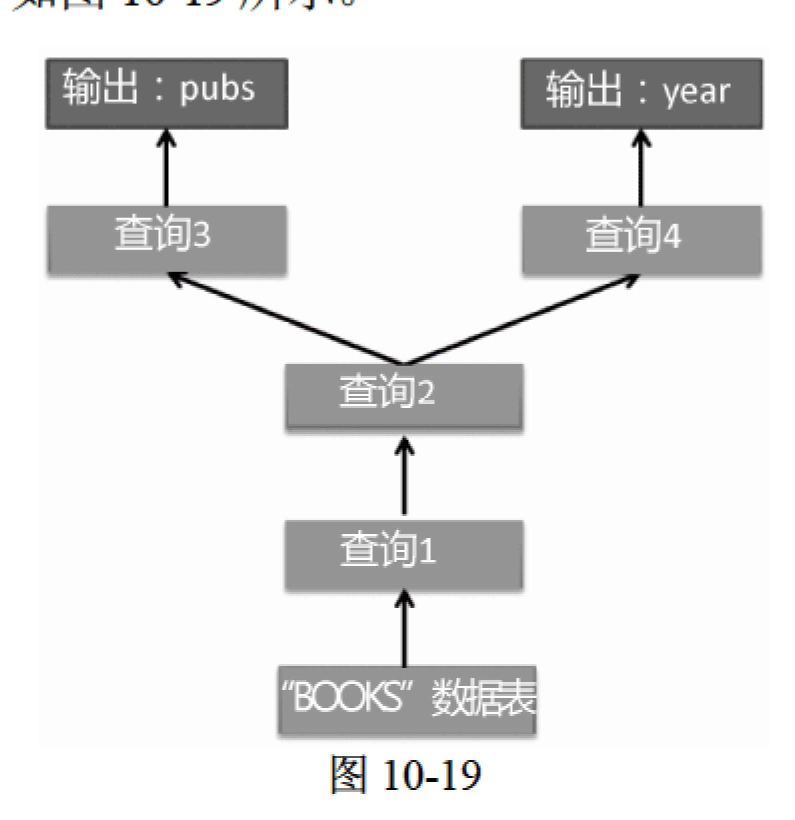


图 10-19

第一个查询我们从“BOOKS”表中取到了发行书数量达到一定要求(根据输入值 cnt)的出版商 ID；

第二个查询我们依据出版商 ID，从“PUBLISHERS”表和“BOOKS”表中取到书和出版商细节信息；

第三个和第四个查询则对第二个查询取的数据价格和书的数量做了些聚合，第三个的聚合依据出版商“PUBS”，第四个的聚合依据发行年份“YEAR”。

像这样一个完全使用声明式构架构造(declarative constructs)的存储过程的例子可以完整地被转换成数据流图，如上图 10-6，和 SQL 查询类似，这样的数据流图可在执行前再次被分析和优化。

对于更复杂的情况，比如主存储过程调用其他的子存储过程，对应到数据流图的概念，我们可以把其中的数据流图中的一个节点可以看成另外一个子数据流图，节点输入输出会相应贯穿连通起来，就和 ABAP 的函数模型或者 ABAP 类方法调用一样。对总体数据流图，SAP HANA 会做分析和优化。

创建完存储过程后，我们还需要测试和调用，看存储过程是不是能正常工作。接下来我们看看那存储过程的调用。

六、存储过程的调用

我们先看下存储过程的调用的几种方式：

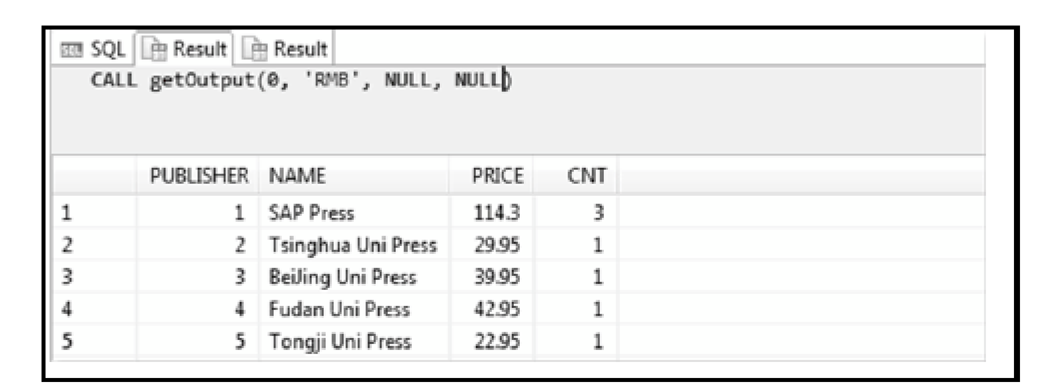
- 从客户端调用(Procedure Called From Client)

语法：CALL<proc_name>(<param_list>)[WITH OVERVIEW][IN DEBUG MODE]

例如调用刚才在第一个存储过程的例子中创建的存储过程——“getOutput”：

```
CALL getOutput(0, 'RMB', NULL, NULL);
```

我们会得到两个结果集，结果集 1 为按照出版商对书价求和(见图 10-20)。


SQL | Result | Result

CALL getOutput(0, 'RMB', NULL, NULL)

	PUBLISHER	NAME	PRICE	CNT
1	1	SAP Press	114.3	3
2	2	Tsinghua Uni Press	29.95	1
3	3	BeiJing Uni Press	39.95	1
4	4	Fudan Uni Press	42.95	1
5	5	Tongji Uni Press	22.95	1

图 10-20

结果集 2 为按照年份求和(见图 10-21)。

SQL | Result | Result

CALL getOutput(0, 'RMB', NULL, NULL)

	YEAR	PRICE	CNT
1	2006	39.8	1
2	2008	93.89	2
3	2012	24.55	1
4	2011	65.7	2
5	2009	29.95	1
6	2010	42.95	1

图 10-21

- 内部调用(Internal Procedure Call)

一个存储过程调用另外一个存储过程：

```
CALL <proc_name > (<param_list>)
```

比如在一个存储过程中可以直接调用另外一个存储过程：

```
CALL sub_procedure (:lt_temp, lt_output_table);
```

- 命名参数调用(Call with Named Parameters)

如果调用存储过程的时候指明参数名称，那么参数的顺序可以忽略，以下语句结果是一样的。




```
CALL sub_procedure (para1_in=>1, para2_in=>2, p_output_1=>?)和 CALL
sub_procedure (para2_in=>2, para2_in=>1, p_output_1=>?)
```

七、存储过程的调用——WITH OVERVIEW 参数

大家会注意到调用存储过程 CALL 语句中有 WITH OVERVIEW 参数，WITH OVERVIEW 参数的作用是可以把输出的结果的数据集写入输出参数对应的表中。比如在上小节的例子中，输出参数对应的是两张表，名称为 op_publishers 和 op_years，那我们就使用如下语句：

```
CALL getOutput(0, 'EUR', op_publishers, op_years) WITH OVERVIEW;
```

这条语句的执行结果就会把输出结果集 1 写入到 op_publishers 这张表中，把结果集 2 写入到 op_years 这张表中。

下面是具体的例子：

我们需要先准备两张表，“op_publishers”和“op_years”，这两张表的作用是将输出的结果存储下来，因此表的结构要和输出数据类型一致。

创建表的代码：

```
CREATE TABLE op_publishers(
   publisher INTEGER,
   name VARCHAR(50),
   price DECIMAL,
   cnt INTEGER);

CREATE TABLE op_years(
   year VARCHAR(4),
   price DECIMAL,
   cnt INTEGER);
```

创建好后，我们可以看到刚建好的表为空表，无任何记录(见 10-22)。

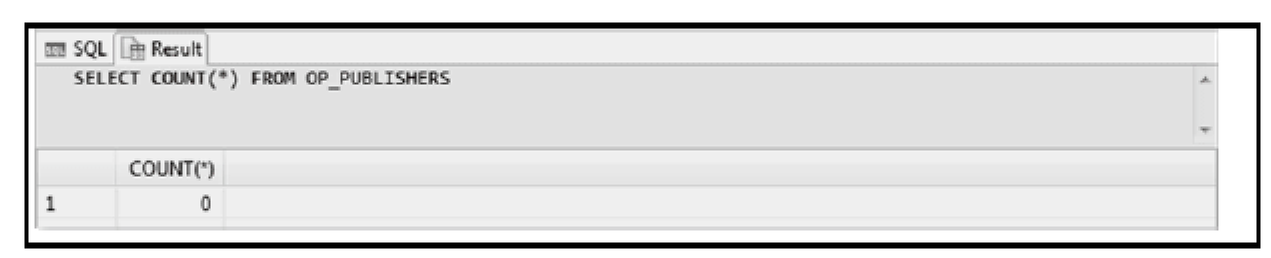


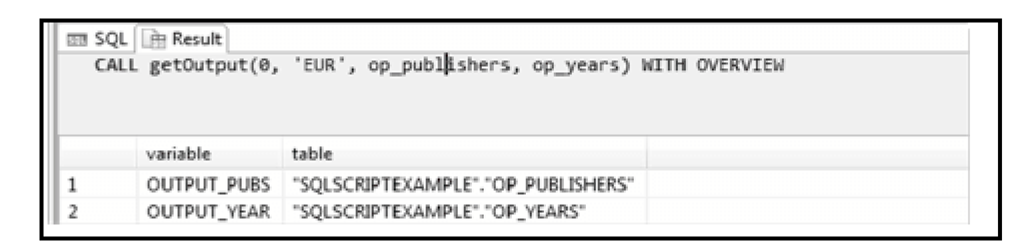
图 10-22



接下来我们调用第一个存储过程例子的存储过程，并加入 WITH OVERVIEW 参数：

```
CALL getOutput(0, 'RMB', op_publishers, op_years) WITH OVERVIEW;
```

运行结果如图 10-23 所示。



```
CALL getOutput(0, 'EUR', op_publishers, op_years) WITH OVERVIEW
```

	variable	table
1	OUTPUT_PUBS	"SQLSCRIPTEXAMPLE"."OP_PUBLISHERS"
2	OUTPUT_YEAR	"SQLSCRIPTEXAMPLE"."OP_YEARS"

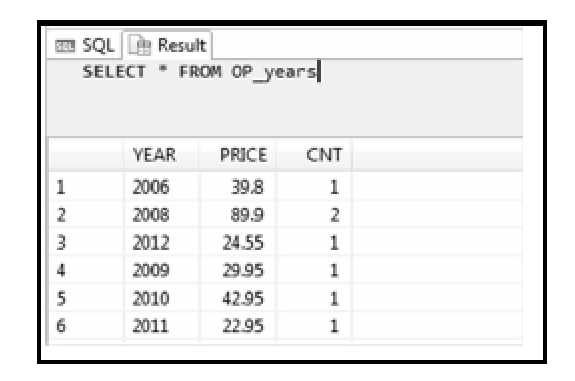
图 10-23

从上图我们能看到，变量“OUTPUT_PUBS”的返回结果被输入到表“op_publishers”中(见图 10-24)；变量“OUTPUT_YEAR”的返回结果被输入到表“op_years”中(见图 10-25)，那我们返回到这两张表，查看下具体内容。

```
SELECT * FROM OP_PUBLISHERS
```

	PUBLISHER	NAME	PRICE	CNT
1	1	SAP Press	114.3	3
2	2	Tsinghua Uni Press	29.95	1
3	3	Beijing Uni Press	39.95	1
4	4	Fudan Uni Press	42.95	1
5	5	Tongji Uni Press	22.95	1

图 10-24



```
SELECT * FROM OP_years
```

	YEAR	PRICE	CNT
1	2006	39.8	1
2	2008	89.9	2
3	2012	24.55	1
4	2009	29.95	1
5	2010	42.95	1
6	2011	22.95	1

图 10-25

可以看出，这两张表已经都有返回结果数据存入。需要指出的是，如果调用时，输出没有给定表名，而是用的 NULL，数据会被写入系统生成的表中，例子如下。

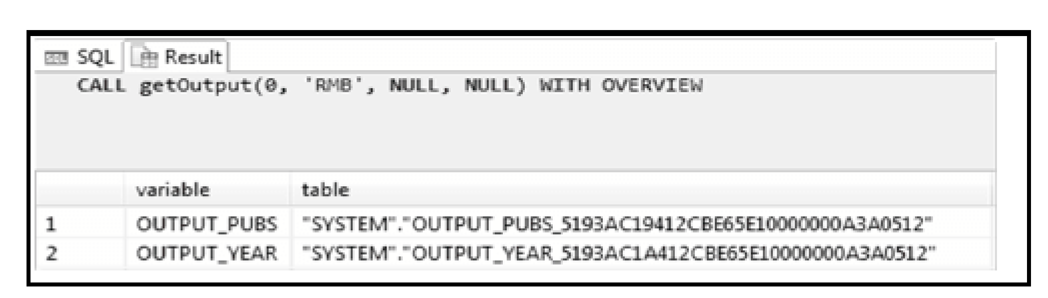
首先我们用“TRUNCATE”命令将上面两个表的内容清空：

```
TRUNCATE TABLE op_publishers;
TRUNCATE TABLE op_years;
```

接下来还是调用相同的存储过程，但是不指定表名称：

```
CALL getOutput(0, 'RMB', NULL, NULL) WITH OVERVIEW;
```

运行结果如图 10-26 所示。



```
CALL getOutput(0, 'RMB', NULL, NULL) WITH OVERVIEW
```

	variable	table
1	OUTPUT_PUBS	"SYSTEM"."OUTPUT_PUBS_5193AC19412CBE65E10000000A3A0512"
2	OUTPUT_YEAR	"SYSTEM"."OUTPUT_YEAR_5193AC1A412CBE65E10000000A3A0512"

图 10-26

由上图可以看出，系统随机生成了两个数据表来存放变量“OUTPUT_PUBS”和变量“OUTPUT_YEAR”的返回结果。在系统中查看随机表内容，可得到如图 10-27

所示结果。

SQL | Result

SELECT * FROM "SYSTEM"."OUTPUT_PUBS_5193AC19412CBE65E10000000A3A0512"

	PUBLISHER	NAME	PRICE	CNT
1	1	SAP Press	114.3	3
2	2	Tsinghua Uni Press	29.95	1
3	3	BeiJing Uni Press	39.95	1
4	4	Fudan Uni Press	42.95	1
5	5	Tongji Uni Press	22.95	1

图 10-27

八、使用 WITH RESULT VIEW 参数创建存储过程

在创建存储过程时，我们可以加入 WITH RESULT VIEW 参数来在创建存储过程时就指定将输出结果写入“新生成”的列视图中。我们看如下的例子，这是个根据书籍价格阶梯打折的存储过程。

首先我们定义要用到的数据表类型。

```
CREATE TYPE tt_sales_books AS TABLE (
   title VARCHAR(50),
   price DECIMAL(5, 2),
   crcy VARCHAR(3))
```

接下来我们创建子存储过程，这个子存储过程可以在今后被其他存储过程调用。其本身的目的是输入总的书籍列表，然后输出价格打折后的书籍列表。

```
CREATE PROCEDURE addDiscount( IN it_books tt_sales_books,
                              OUT ot_books tt_sales_books)
LANGUAGE SQLSCRIPT READS SQL DATA AS
BEGIN
  ot_Books = SELECT title,
                  CASE WHEN price > 30 THEN (price * 0.5)
                  ELSE CASE WHEN price > 20 THEN (price * 0.7)
                  ELSE (price * 0.9)
                  END
             END AS price, crcy
             FROM :it_books;
END;
```

然后我们创建主存储过程，这个存储过程根据用户指定的某个价格把书籍分为两类，“expensive books”和“cheap books”。对于“expensive books”根据上面的子存储过程处理后会有一定的折扣计算，对于“cheap books”直接输出。

```
CREATE PROCEDURE getSalesBooks( IN minPrice DECIMAL(5, 2),
                              IN currency VARCHAR(3),
                                IN it_books books,
                                OUT ot_sales tt_sales_books)
LANGUAGE SQLSCRIPT READS SQL DATA WITH RESULT VIEW addDiscount_RET AS
BEGIN
  lt_expensive_books = SELECT title, price, crcy
                       FROM :it_books
                       WHERE price > :minPrice
                       AND crcy = :currency;

  CALL addDiscount(:lt_expensive_books, lt_on_sale);

  lt_cheap_books = SELECT title, price, crcy
                   FROM :it_books
                   WHERE price <= :minPrice
                   AND crcy = :currency;
  ot_sales = CE_UNION_ALL(:lt_on_sale, :lt_cheap_books);
END;
```

请注意这段代码标灰的部分，在这里我们加入了 WITH RESULT VIEW 参数，调用的列视图为“addDiscount_RET”。我们在 SQL Console 中调用此“getSalesBooks ”存储过程时，相应的列视图会被生成并填充数据。调用的同时，我们也使用上面介绍过的 WITH OVERVIEW 参数。

先为 WITH OVERVIEW 参数准备 TABLE。

```
CREATE INSERT ONLY COLUMN TABLE op_sales_books(
   title VARCHAR(50),
   price DECIMAL(5, 2),
   crcy VARCHAR(3));
```

然后我们调用这个存储过程：

```
CALL getSalesBooks(25, 'RMB' , books, op_sales_books) WITH OVERVIEW;
```

运行结果如图 10-28 所示。

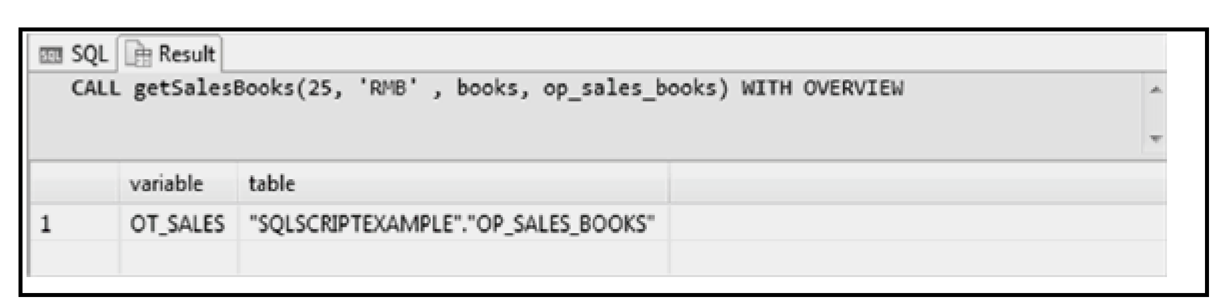


图 10-28

我们返回到“Column Views”节点查看相应的视图是否生成，同时查看结果(见图 10-29)。

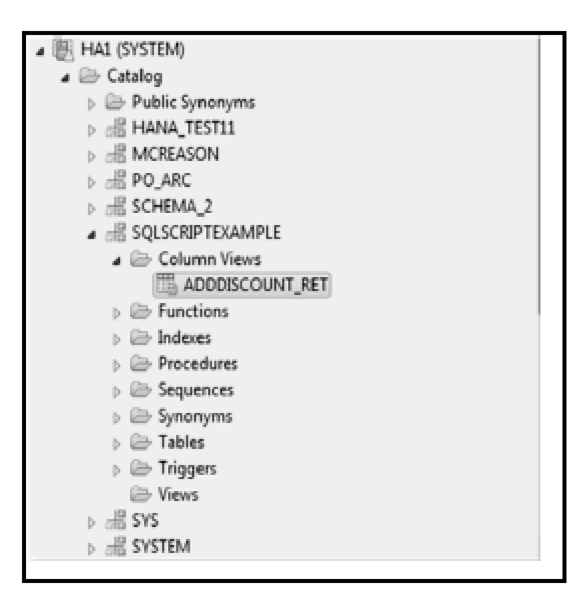


图 10-29

由图 10-29 能够看出，在我们默认的 Schema 下面，相应的“Column View”已经生成。通过如下 SQL 语句，我们同样也可从生成的“Column View”中读取数据，这个在写 SQL Script 时非常有用，它实现了一个存储过程 A 的数据输出给另外一个存储过程 B 使用。

```
SELECT * FROM addDiscount_RET WITH PARAMETERS (
'placeholder' = ('$$minprice$$', '25'),
              'placeholder' = ('$$currency$$', 'RMB'),
              'placeholder' = ('$$it_books$$', 'books'),
              'placeholder' = ('$$ot_sales$$', 'op_sales_books'));
```

运行结果如图 10-30 所示。

接下来看看相应的输出的表“op_sales_books”是否有插入数据：

```
SELECT * FROM OP_SALES_BOOKS;
```

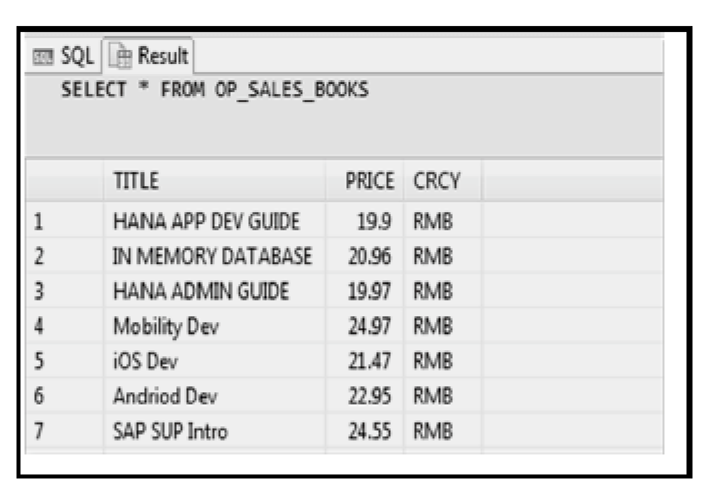
运行结果如图 10-31 所示。

SQL | Result

```
SELECT * FROM addDiscount_RET WITH PARAMETERS (
 'placeholder' = ('$$minprice$$', '25'),
 'placeholder' = ('$$currency$$', 'RMB'),
 'placeholder' = ('$$it_books$$', 'books'),
 'placeholder' = ('$$ot_sales$$', 'op_sales_books'))
```

	TITLE	PRICE	CRCY
1	HANA APP DEV GUIDE	19.90	RMB
2	IN MEMORY DATABASE	20.96	RMB
3	HANA ADMIN GUIDE	19.97	RMB
4	Mobility Dev	24.97	RMB
5	iOS Dev	21.47	RMB
6	Andriod Dev	22.95	RMB
7	SAP SUP Intro	24.55	RMB

图 10-30


SQL | Result

```
SELECT * FROM OP_SALES_BOOKS
```

	TITLE	PRICE	CRCY
1	HANA APP DEV GUIDE	19.9	RMB
2	IN MEMORY DATABASE	20.96	RMB
3	HANA ADMIN GUIDE	19.97	RMB
4	Mobility Dev	24.97	RMB
5	iOS Dev	21.47	RMB
6	Andriod Dev	22.95	RMB
7	SAP SUP Intro	24.55	RMB

图 10-31

数据已经被写入，成功运行。

第三节　SQL Script 技术要点

一、变量和引用

1. 数据表变量(Table Variable)

在 SAP HANA SQL Script 中，我们通过“=”将 SQL 语句的查询结果直接赋值给数据表变量，例如前面例子中的：

```
lt_expensive_books = SELECT title, price, crcy FROM :it_books WHERE price > :minPrice AND crcy = :currency;
```

“lt_expensive_books”即为数据表变量，我们通过“=”符号将 SQL 查询语句的输出结果集赋值给数据表变量。

2. 标量变量 (Scalar Variable)

标量变量用于表示各个大小固定的数据对象(如整数)。在 SAP HANA SQL Script 中，我们通过“:=”符号为标量变量赋值，通过“:”符号来声明标量变量，例如“:minPrice”、“:currency”都是引用的输入标量变量。下面我们再来看一个最简单的引用和赋值的例子：

```
create procedure scalar_proc(IN a bigint, OUT b bigint) LANGUAGE SQLSCRIPT AS
BEGIN
    init_proc();
    ins_msg_proc('a = ' || :a);
    b := :a + 1;
    ins_msg_proc('b = ' || :b);
END;
```

我们在这个存储过程中定义了“a”为输入参数，“b”为输出参数。通过“b:= :a + 1”语句我们赋予“b”的值为输入参数“a”的值加“1”。“||”操作符的作用是拼接其前后的字符，我们会在接下来的消息结果中看到拼接的结果。下面我们调用这个存储过程，并给“a”赋值为“1”：

```
CALL SCALAR_PROC(1, ?);
```

查看执行结果(见图 10-32、图 10-33)。

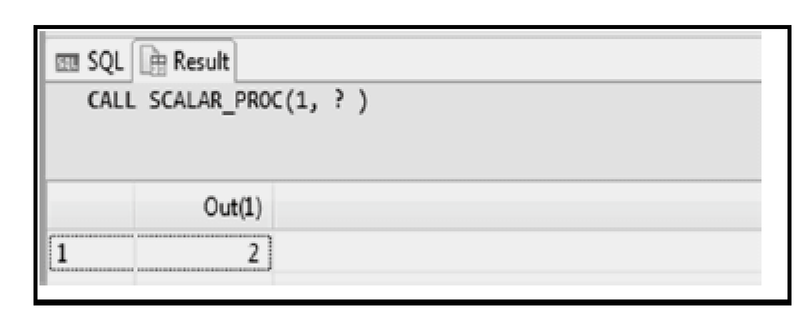


图 10-32

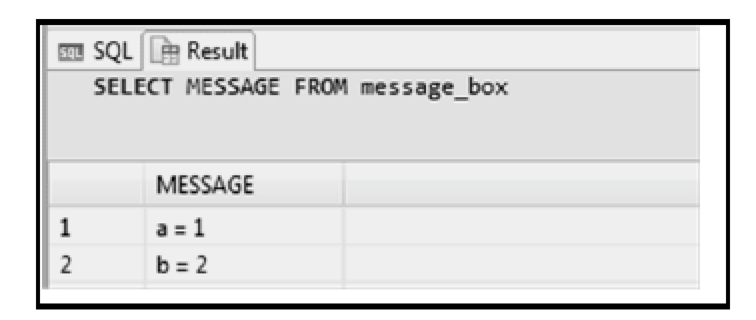


图 10-33

另外，我们也可以在“AS”后面直接声明本地的标量变量，并给出定义类型和初始值。还是通过“:=”或者“SELECT INTO”为变量赋值；通过“:”引用变量。

```
CREATE PROCEDURE sql_proc LANGUAGE SQLSCRIPT AS
    v_count   INT := 0;
    v_isbn    VARCHAR(20);
    v_title   VARCHAR(50) := '';
    v_price   decimal(5,2) := 0;
    v_crcy    VARCHAR(3) := 'XXX';
BEGIN
    init_proc();

v_isbn := '978-3-8266-1664-8';

-- SELECT INTO VARIABLE
    SELECT isbn, title, price, crcy INTO v_isbn, v_title, v_price, v_crcy
FROM books WHERE isbn = :v_isbn;

-- INSERT MESSAGE WITH REF
ins_msg_proc(:v_title || ' identified by isbn ' || :v_isbn ||
' costs ' || :v_price || ' ' || :v_crcy);

    SELECT COUNT(*) INTO v_count FROM books;
    ins_msg_proc('table book has ' || :v_count || ' records');

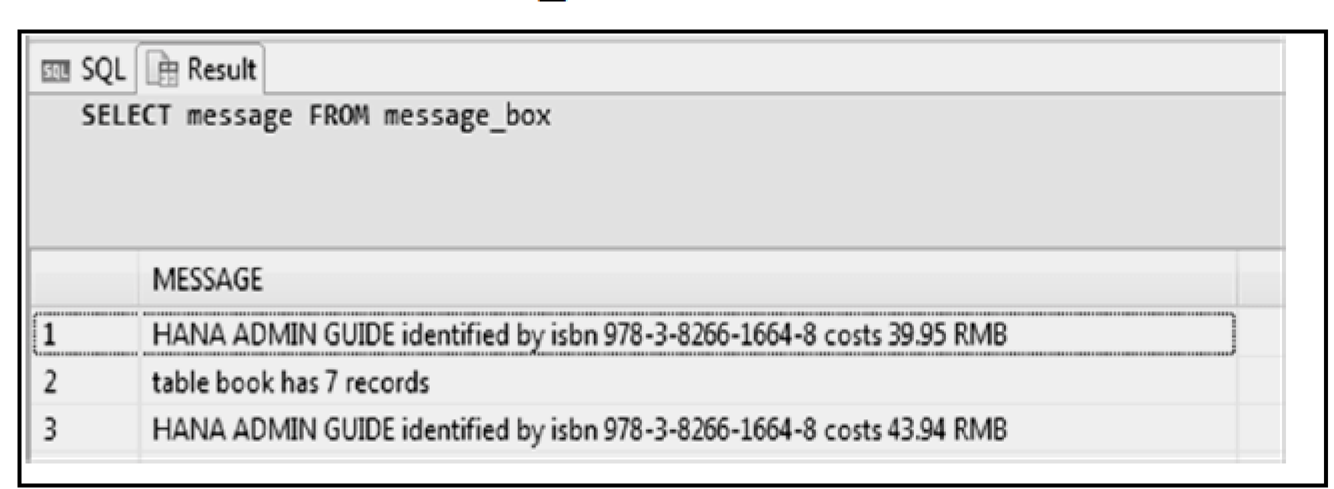
    -- UPDATE
    UPDATE books SET price = price * 1.1 WHERE isbn = :v_isbn;

SELECT isbn, title, price, crcy INTO v_isbn, v_title, v_price, v_crcy
FROM books WHERE isbn = :v_isbn;
ins_msg_proc(:v_title || ' identified by isbn ' || :v_isbn ||
' costs ' || :v_price || ' ' || :v_crcy);
END;
```

查看输出结果(见图 10-34)：

```
CALL sql_proc();

SELECT message FROM message_box;
```



	MESSAGE
1	HANA ADMIN GUIDE identified by isbn 978-3-8266-1664-8 costs 39.95 RMB
2	table book has 7 records
3	HANA ADMIN GUIDE identified by isbn 978-3-8266-1664-8 costs 43.94 RMB

图 10-34

二、逻辑控制

前面我们讲述了变量及其引用，本小节中我们介绍下 SAP HANA SQL Script 中的逻辑控制语句，比如：

```
IF THEN ELSE
WHILE LOOP
FOR LOOP
BREAK & CONTINUE
```

我们先从 IF THEN ELSE 开始。

- IF THEN ELSE

语法：

```
IF <bool_expr1>
THEN <then_stmts1>
[{ELSEIF <bool_expr2>
THEN <then_stmts2>}...]
[ELSE <else_stmts3>]
END IF;
```

举个简单的例子，如果没找到相应的书号(ISBN)，则新插入数据，找到了就更新价格。

```
CREATE PROCEDURE if_proc (IN v_isbn VARCHAR(20)) LANGUAGE SQLSCRIPT
AS
    found INT := 1;
```

```
BEGIN
    init_proc();
    SELECT count(*) INTO found FROM books WHERE isbn = :v_isbn;
    IF :found = 0 THEN
INSERT INTO books VALUES (:v_isbn, 'In-Memory Data Management v3',
1, 1, '2011', 42.99, 'RMB');
       ELSE
          UPDATE books SET price = 42.88 WHERE isbn =:v_isbn;

    END IF;

END;
```

- WHILE LOOP

语法：

```
WHILE <condition> DO
<proc_stmts>
END WHILE
```

接下来我们来看一个当满足 WHILE 条件进行 DO 循环的例子：

```
CREATE PROCEDURE while_proc LANGUAGE SQLSCRIPT AS
    v_index1 INT := 0;
    v_index2 INT := 0;
    v_msg    VARCHAR(200) := '';
BEGIN
    init_proc();

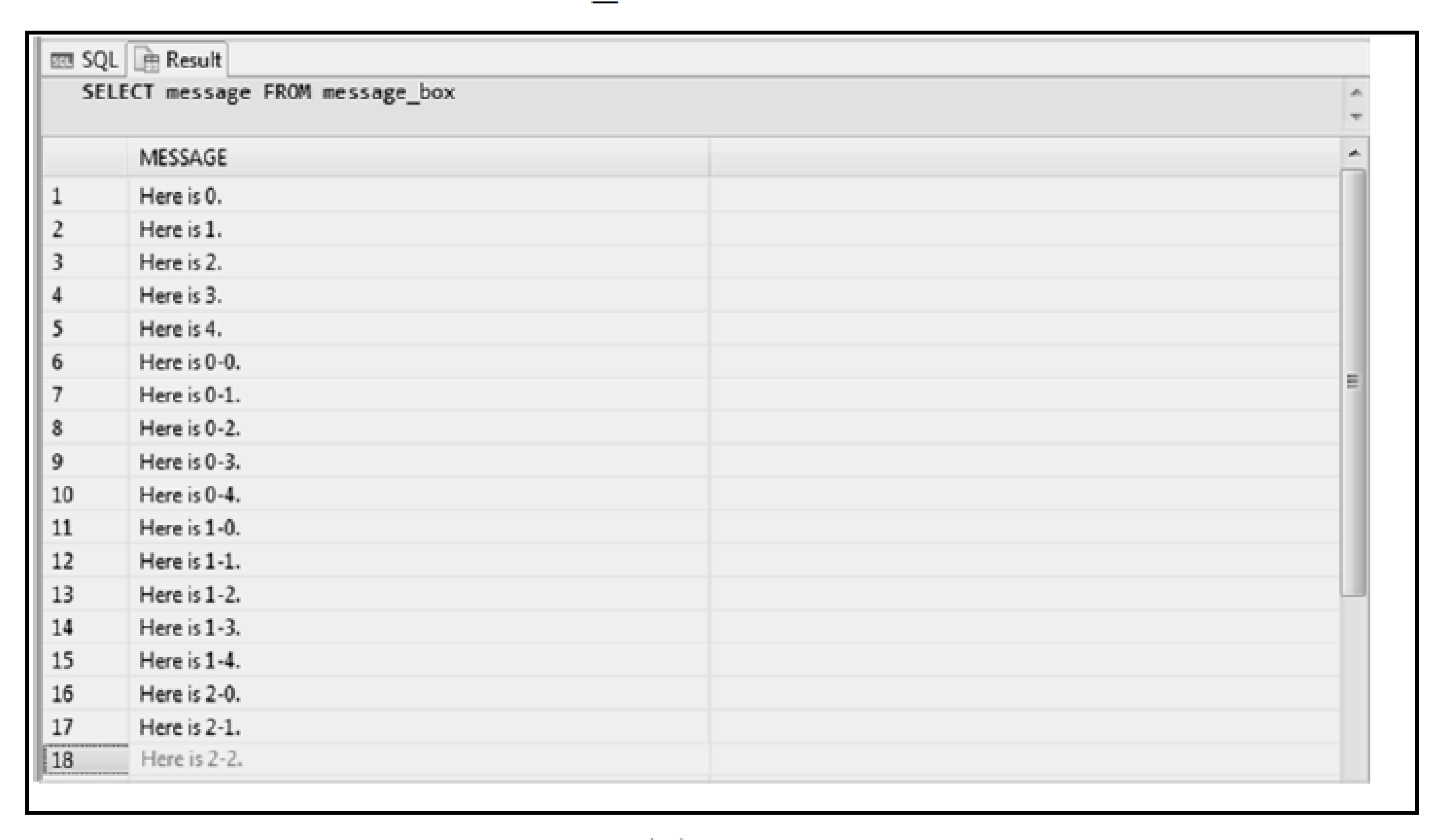
-- 简单的"While-DO"语句
    WHILE :v_index1 < 5 DO
        v_msg := 'Here is ' || :v_index1 || '.';
        ins_msg_proc(:v_msg);
        v_index1 := :v_index1 + 1;
    END WHILE;

-- 嵌套的"While-DO"语句
v_index1 := 0;
    WHILE :v_index1 < 5 DO
        v_index2 := 0;
        WHILE :v_index2 < 5 DO
            v_msg := 'Here is '|| :v_index1 || '-' || :v_index2 ||
'.';
```

```
            ins_msg_proc(:v_msg);
            v_index2 := :v_index2 + 1;
        END WHILE;
        v_index1 := :v_index1 + 1;
    END WHILE;
END;
```

在这个存储过程里面，我们先在“AS”后面声明两个本地计数变量(“v_index1”和“v_index2”)，其初始值均为 0。在第一个简单的“WHILE DO”语句里，往“MESSAGE”里面增加相应含有 INDEX 的消息，再递增“INDEX1”，继续执行。在第二个嵌套“WHILE DO”语句里，我们外层递增“INDEX1”，里层递增“INDEX1”，往 message 里面增加相应含有 INDEX 的消息。通过 SELECT FROM message 语句我们能看到这个 WHILE 循环的结果(见图 10-35)：

```
CALL while_proc();
SELECT message FROM message_box;
```


SELECT message FROM message_box

	MESSAGE
1	Here is 0.
2	Here is 1.
3	Here is 2.
4	Here is 3.
5	Here is 4.
6	Here is 0-0.
7	Here is 0-1.
8	Here is 0-2.
9	Here is 0-3.
10	Here is 0-4.
11	Here is 1-0.
12	Here is 1-1.
13	Here is 1-2.
14	Here is 1-3.
15	Here is 1-4.
16	Here is 2-0.
17	Here is 2-1.
18	Here is 2-2.

图 10-35

下面这个例子中，我们综合一下 WHILE 和 IF ELSE 控制语句的使用：

```
CREATE PROCEDURE upsert_proc (IN v_isbn VARCHAR(20)) LANGUAGE
SQLSCRIPT
AS found INT := 1;
BEGIN
    init_proc();
    WHILE :found <> 0 DO
        SELECT count(*) INTO found FROM books WHERE isbn = :v_isbn;
        IF :found IS NULL THEN
```

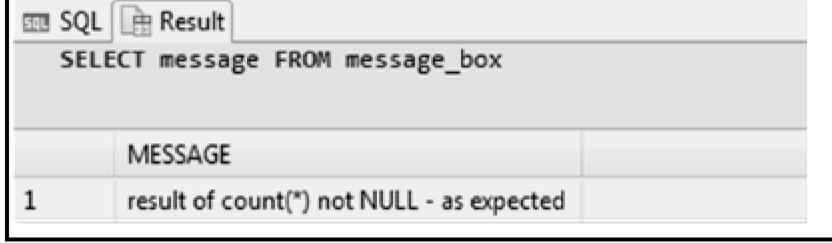
```
            ins_msg_proc('result of count(*) cannot be NULL');
          ELSE
           ins_msg_proc('result of count(*) not NULL - as expected');
      END IF;

          IF :found = 0 THEN
            INSERT  INTO  books  VALUES  (:v_isbn,  'In-Memory  Data
Management V2', 1, 1, '2011', 42.75, 'RMB');
          END IF;
      END WHILE;
  END;
```

使用不存在的 ISBN 号调用 UPSERT 存储过程(见图 10-36、图 10-37)。

```
call upsert_proc('978-3-642-19362-0');

SELECT message FROM message_box;
```



SQL | Result

SELECT message FROM message_box

	MESSAGE
1	result of count(*) not NULL - as expected

图 10-36

```
SELECT * FROM books;
```

SQL | Result

SELECT * FROM books

	ISBN	TITLE	PUBLISHER	EDITION	YEAR	PRICE	CRCY
1	978-3-486-57690-0	HANA APP DEV GUIDE	1	6	2006	39.8	RMB
2	978-3-86894-012-1	IN MEMORY DATABASE	2	3	2009	29.95	RMB
3	978-3-8266-1665-8	Mobility Dev	1	3	2008	49.95	RMB
4	978-3-8266-1665-9	iOS Dev	4	3	2010	42.95	RMB
5	978-3-8266-1665-2	Andriod Dev	5	3	2011	22.95	RMB
6	978-3-8266-1665-4	SAP SUP Intro	1	3	2012	24.55	RMB
7	978-3-8266-1664-8	HANA ADMIN GUIDE	3	3	2008	43.94	RMB
8	978-3-642-19362-0	In-Memory Data Management V2	1	1	2011	42.75	RMB

图 10-37

- FOR LOOP

语法：

```
FOR <loop-var> IN [REVERSE] <start_value> .. <end_value> DO
<proc_stmts>
END FOR
```

我们同样用上面两层索引的例子，只不过这次我们使用 FOR 循环来完成

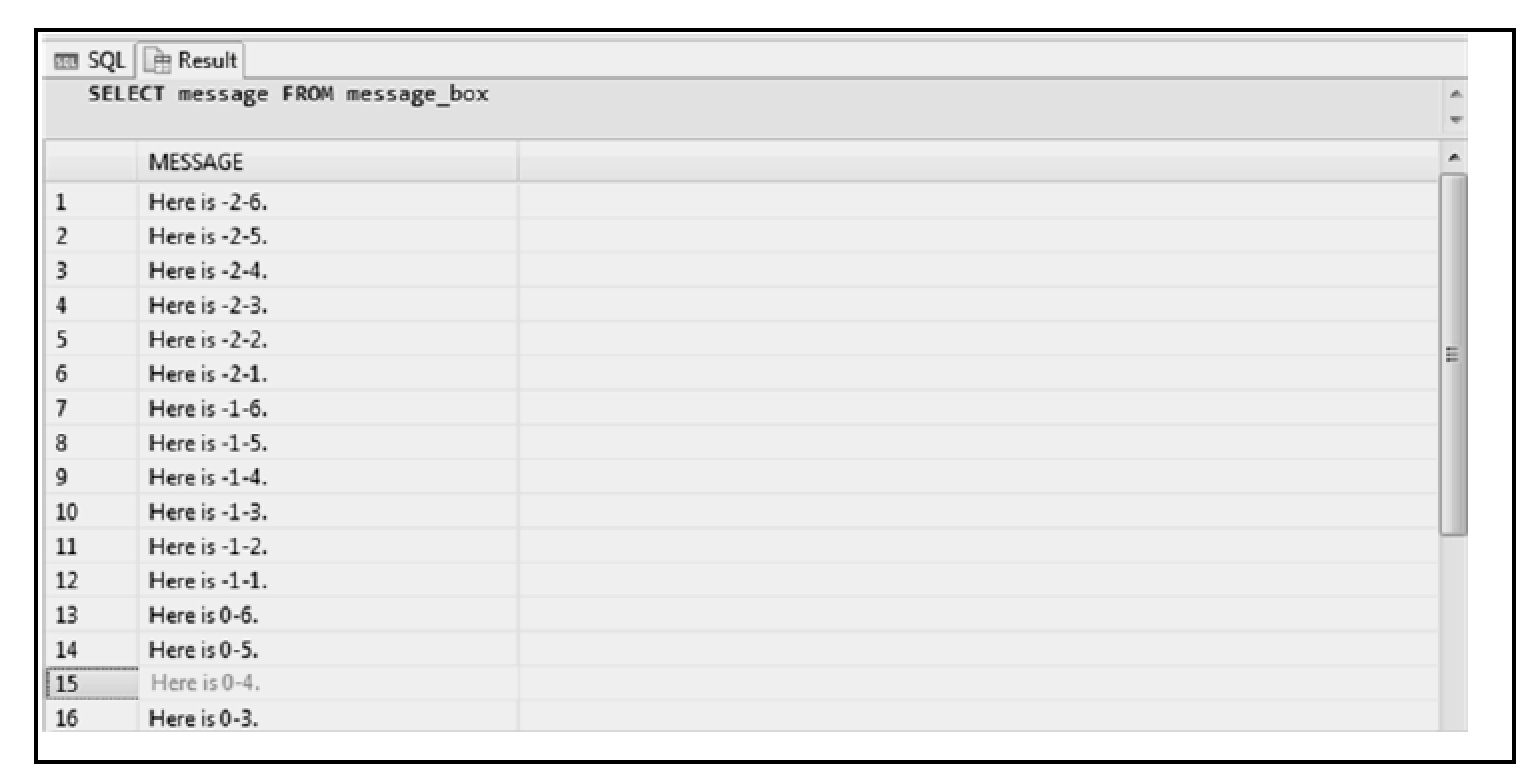


message 的写入：

```
CREATE PROCEDURE for_proc LANGUAGE SQLSCRIPT AS
    v_index1 INT;
    v_index2 INT;
    v_msg     VARCHAR(200);
BEGIN
    init_proc();

    FOR v_index1 IN -2 .. 2 DO
        FOR v_index2 IN REVERSE 1 .. 6 DO
            v_msg := 'Here is '|| :v_index1 || '-' || :v_index2 || '.';
            ins_msg_proc(:v_msg);
        END FOR;
    END FOR;
END;
CALL for_proc();
SELECT message FROM message_box;
```

结果如图 10-38 所示。


SQL　Result

SELECT message FROM message_box

	MESSAGE
1	Here is -2-6.
2	Here is -2-5.
3	Here is -2-4.
4	Here is -2-3.
5	Here is -2-2.
6	Here is -2-1.
7	Here is -1-6.
8	Here is -1-5.
9	Here is -1-4.
10	Here is -1-3.
11	Here is -1-2.
12	Here is -1-1.
13	Here is 0-6.
14	Here is 0-5.
15	Here is 0-4.
16	Here is 0-3.

图 10-38

三、游标

游标可以用来从查询的结果集中取出某一行数据。当定义游标的时候，查询往往和此游标绑定。如果还为游标定义了参数，当打开游标的时候还需提供相应参数值。

CURSOR 相关语法：

```
CURSOR <cursor_name> [({<param_def>{,<param_def>} ...)]
```

```
FOR <select_stmt>:
```

下面我们通过实际的例子来演示下游标的打开(Open)、关闭(Close)、提取(Fetch)：

```
CREATE PROCEDURE cursor_proc LANGUAGE SQLSCRIPT AS
    v_isbn     VARCHAR(20);
    v_title    VARCHAR(50) := '';
    v_price    decimal(5,2) := 0;
    v_crcy     VARCHAR(3) := 'RMB';
    v_msg      VARCHAR(200);
CURSOR c_cursor1 (v_isbn VARCHAR(20)) FOR SELECT isbn, title, price,
crcy FROM books WHERE isbn = :v_isbn ORDER BY isbn;
BEGIN
init_proc();

-- 检查游标状态是否为关闭
   IF c_cursor1::ISCLOSED     THEN
          ins_msg_proc('OK: cursor not open');
   ELSE
          ins_msg_proc('WRONG: cursor open');
   END IF;

-- 使用参数打开游标
    OPEN c_cursor1('978-3-86894-012-1');
   IF c_cursor1::ISCLOSED     THEN
          ins_msg_proc('WRONG: cursor not open');
   ELSE
          ins_msg_proc('OK: cursor open');
   END IF;

-- 取得数据，如未取得则返回错误信息
    FETCH c_cursor1 INTO v_isbn, v_title, v_price, v_crcy;
    IF c_cursor1::NOTFOUND THEN
        ins_msg_proc('WRONG: cursor contains no valid data');
   ELSE
          ins_msg_proc('OK: cursor contains valid data');
ins_msg_proc(:v_title || ' identified by isbn ' || :v_isbn || '
costs ' || :v_price || ' ' || :v_crcy);
   END IF;
   IF c_cursor1::ISCLOSED
   THEN
        ins_msg_proc('WRONG: cursor not open');
   ELSE
          ins_msg_proc('OK: cursor open');
   END IF;
```

```
-- 关闭游标
    CLOSE c_cursor1;
    IF c_cursor1::ISCLOSED
    THEN
        ins_msg_proc('OK: cursor not open');
    ELSE
        ins_msg_proc('WRONG: cursor open');
    END IF;
END;
call cursor_proc();
select message from message_box;
```

结果如图 10-39 所示。

SQL Result

select message from message_box

	MESSAGE
1	OK: cursor not open
2	OK: cursor open
3	OK: cursor contains valid data
4	IN MEMORY DATABASE identified by isbn 978-3-86894-012-1 costs 29.95 RMB
5	OK: cursor open
6	OK: cursor not open

图 10-39

在实际编程中，我们可以通过语句“FOR cur_row AS c_cursor1 DO”对游标的数据集合做 LOOP 循环，来取得每一行的数据，例如：

```
CREATE PROCEDURE foreach_proc() LANGUAGE SQLSCRIPT AS
    v_isbn    VARCHAR(20) := '';
    CURSOR c_cursor1 FOR
       SELECT isbn, title, price, crcy FROM books
       ORDER BY isbn;
BEGIN
    init_proc();
    FOR cur_row AS c_cursor1 DO
        ins_msg_proc('book title is: ' || cur_row.title);
    END FOR;
END;

call foreach_proc();
select message from message_box;
```

结果如图 10-40 所示。



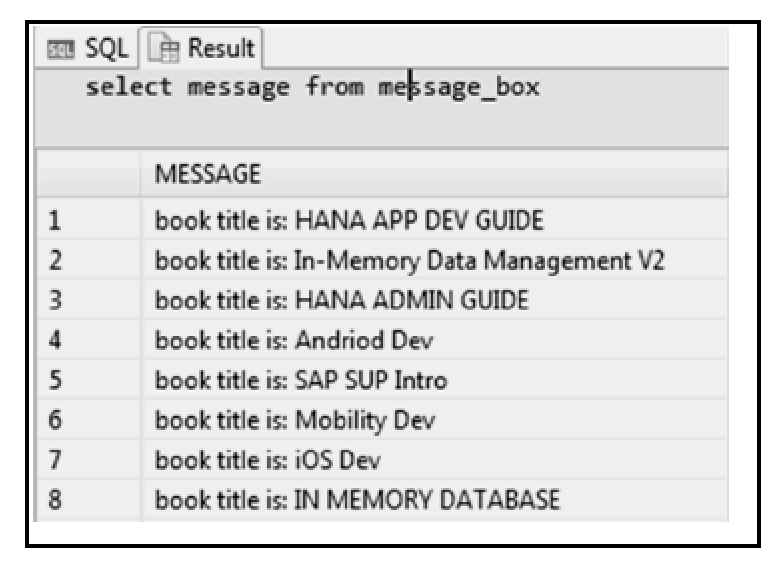


图 10-40

四、动态 SQL

通过语句“EXEC”我们可以执行以字符串形式拼成的 SQL 语句。

语法：`EXEC '<sql-statement>'`

实例：

```
CREATE PROCEDURE dynamic_sql_proc (IN iv_table VARCHAR(100) )
LANGUAGE SQLSCRIPT AS
    v_sql1    VARCHAR(1024);
    v_sql2    VARCHAR(1024);
    v_msg     VARCHAR(200);
BEGIN
    init_proc();

    v_sql1 := 'INSERT INTO "MESSAGE_BOX" VALUES (''1 message from
Dynamic SQL'', SYSTIMESTAMP)';
    EXEC :v_sql1;

    v_sql2 := 'INSERT INTO message_box VALUES (''2 message from
Dynamic SQL'', SYSTIMESTAMP)';
    EXEC :v_sql2;

    v_sql1 := '3 message from Dynamic SQL';
    EXEC 'INSERT INTO message_box VALUES (''' || :v_sql1 || ''',
SYSTIMESTAMP)';
END;
```

五、计算引擎函数

在前面 SAP HANA 存储过程简介中我们提到，SAP HANA 包含专门用于处理特定事务的引擎，比如 Calculation Engine、Join Engine、OLAP Engine、Row store engine、Column store engine、Text Engine 等。在用 SQL Script 创建存储过程的时候，为了更高效使用计算引擎实现计算，SAP HANA 提供了基于计算引擎的一些函数，我们常称之为“CE Function”。开发人员可以使用 CE Function 直接和计算引擎对话，相应的查询不会触发 SQL processor，使得计算执行更加高效。

这些 CE Function 大概有如下几类：

- 用于读取数据的，比如读取某些表或者试图的数据
  - CE_COLUMN_TABLE
  - CE_JOIN_VIEW
  - CE_OLAP_VIEW
  - CE_CALC_VIEW
- 关系型操作的语句(JOIN/PROJECTION/AGGREGATION/UNION_ALL)
  - CE_JOIN
  - CE_LEFT_OUTER_JOIN
  - CE_RIGHT_OUTER_JOIN
  - CE_PROJECTION
  - CE_CALC
  - CE_AGGREGATION
  - CE_UNION_ALL
- 其他特殊用途的语句
  - CE_VERTICAL_UNION
  - CE_CONVERSION

大家对于 SQL 还比较熟悉，下面我们来看看 SQL 语句映射到这些 CE Function 的例子。

- 从表或者试图读取数据

例 1：读取表的所有字段

```
SQL: ot_books1 = SELECT * FROM books;
CE:  ot_books1 = CE_COLUMN_TABLE("BOOKS");
```



例 2：读取表的某些字段

```
SQL: ot_books2 = SELECT title, price, crcy FROM books;
CE:  ot_books2 = CE_COLUMN_TABLE("BOOKS", ["TITLE","PRICE","CRCY"]);
```

- 从属性视图中读取数据：

```
SQL:  SELECT product_key, product_text, sales FROM product_sales;
CE:   CE_JOIN_VIEW("PRODUCT_SALES", ["PRODUCT_KEY", "PRODUCT_TEXT",
"SALES"]);
```

- 从分析视图中读取数据：

```
SQL:  select dim1, SUM(kf) FROM OLAP_view GROUP BY dim1;
CE:   CE_OLAP_VIEW("OLAP_view", ["DIM1", SUM("KF")]);
```

- 从计算视图中读取数据：

```
SQL:  SELECT cid, cname FROM "_SYS_SS_CE_TESTCECTABLE_RET";
CE:   CE_CALC_VIEW("_SYS_SS_CE_TESTCECTABLE_RET", ["CID", "CNAME"]);
```

接下来让我们来看看用于关系型操作的 SQL 语句转换到 CE Function 的例子。

- JOIN：

```
SQL:  SELECT title, name, P.publisher AS publisher, year
FROM :lt_pubs AS P, :it_books AS B WHERE P.publisher = B.publisher;
CE:   CE_JOIN (:lt_pubs, :it_books,["PUBLISHER"], ["TITLE","NAME","
PUBLISHER","YEAR"]);
```



- Projection：

```
SQL:  SELECT title, price, crcy AS currency FROM :it_books WHERE price > 50;

CE:   CE_PROJECTION (:it_books,["TITLE","PRICE", "CRCY" AS "CURRENC
Y"], '"PRICE" > 50');
```

- Aggregation：

```
SQL:  SELECT COUNT (publisher) AS cnt, year FROM :it_books GROUP BY year;

CE:   CE_AGGREGATION (:it_books, [COUNT ("PUBLISHER") AS "CNT"], ["YEAR"]);
```

- UNION ALL：

```
SQL: SELECT * FROM :lt_books UNION ALL SELECT * FROM :it_audiobooks;
```

```
CE: CE_UNION_ALL (:lt_books, :it_audiobooks);
```

最后我们来看一个使用 CE Function 创建的简单的存储过程的例子。

先准备两张存储测试数据的表：

```
CREATE TABLE  op_proj_books(
   title VARCHAR(50),
   price DECIMAL(5, 2),
   price_vat DECIMAL(5, 2),
   currency VARCHAR(3));

CREATE TABLE op_colt_books(
   title VARCHAR(50),
   price DECIMAL(5, 2),
   price_vat DECIMAL(5, 2),
   crcy VARCHAR(3));
```

我们要创建的存储过程的作用是从“BOOKS”数据表中读取数，并对价格做简单的计算：

```
CREATE PROCEDURE ceProjectBooks( IN it_books books,
                                 OUT ot_books1 tt_proj_books,
                                 OUT ot_books2 tt_colt_books)
LANGUAGE SQLSCRIPT READS SQL DATA AS
BEGIN
  ot_books1 = CE_PROJECTION(:it_books, ["TITLE", "PRICE", CE_CALC('
"PRICE" * 1.5', decimal(5,2)) AS "PRICE_VAT", "CRCY" AS "CURRENCY"]);

  ot_books2 = SELECT title, price, price * 1.5 as price_vat, crcy
              FROM :it_books;
END;
```

对于“ot_books1”我们使用 CE Function 进行数据查询，对于“ot_books2”我们使用普通的 SQL 语句进行同样逻辑的数据查询，调用存储过程并把数据存储到两张输出表中：

```
CALL   ceProjectBooks(BOOKS,   op_proj_books,   op_colt_books)   WITH
OVERVIEW;
SELECT * FROM op_proj_books;
SELECT * FROM op_colt_books;
```

执行上面这两个语句，查看这两数据表中的数据是否相同。

六、存储过程系统属性信息

当存储过程被创建后，系统会在“Catalog”目录中创建相应的存储过程基本信息。主要的信息如表 10-1 所示。

表 10-1

列 名 称	描 述
SCHEMA_NAME	Schema 名称
PROCEDURE_NAME	Procedure 名称
PROCEDURE_OID	Procedure ID
OWNER_OID	所有人 ID
INPUT_PARAMETER_COUNT	输入参数 数目
OUTPUT_PARAMETER_COUNT	输出参数数目
INOUT_PARAMETER_COUNT	输入出参数数目
IS_UNICODE	是否是 UNICODE
DEFINITION	定义，string， 也就是 Procedure 代码
PROCEDURE_TYPE	类型

图 10-41 是通过 SQL 语句查询从“SYS.PROCEDURES”和“PROCEDURE_PARAMAMETERS VIEW”中读取目前我们创建过在“SQLSCRIPTEXAMPLE SCHEMA”下的存储过程的信息。



SQL | Result

```
select * from sys.procedures p, sys.procedure_parameters pp
where p.procedure_oid = pp.procedure_oid
and p.schema_name = 'SQLSCRIPTEXAMPLE'
```

	SCHEMA_NAME	PROCEDURE_NAME	PROCEDURE_OID	INPUT_PARAMETER_COUNT	OUTPUT_PARAMETEI
1	SQLSCRIPTEXAMPLE	CEPROJECTBOOKS	173108	1	
2	SQLSCRIPTEXAMPLE	CEPROJECTBOOKS	173108	1	
3	SQLSCRIPTEXAMPLE	CEPROJECTBOOKS	173108	1	
4	SQLSCRIPTEXAMPLE	GETOUTPUT	172697	2	
5	SQLSCRIPTEXAMPLE	ADDDISCOUNT	172715	1	
6	SQLSCRIPTEXAMPLE	ADDDISCOUNT	172715	1	
7	SQLSCRIPTEXAMPLE	GETSALESBOOKS	172739	3	
8	SQLSCRIPTEXAMPLE	GETSALESBOOKS	172739	3	
9	SQLSCRIPTEXAMPLE	GETSALESBOOKS	172739	3	
10	SQLSCRIPTEXAMPLE	GETSALESBOOKS	172739	3	
11	SQLSCRIPTEXAMPLE	DYNAMIC_SQL_PROC	173100	1	
12	SQLSCRIPTEXAMPLE	SCALAR_PROC	172899	1	
13	SQLSCRIPTEXAMPLE	SCALAR_PROC	172899	1	
14	SQLSCRIPTEXAMPLE	UPSERT_PROC	172996	1	
15	SQLSCRIPTEXAMPLE	IF_PROC	173004	1	

图 10-41

第四节　SQL Script 开发注意事项

前面我们介绍了 SAP HANA SQL Script 的一些基本语法，接下来我们了解下使用 SQL Script 进行开发需要注意的一些事项。

一、充分使用 HANA 内置引擎

大家在前面第八章已经看到，SAP HANA 提供了强有力的建模环境，当我们需要设计一个 SAP HANA 应用时，应先分析数据特性，根据实际情况看是用建模就足够了，还是需要使用 SQL Script 实现复杂计算逻辑。比如分析数据后发现，如果应用涉及很多表和复杂的连接，那我们可以使用图形化方式的属性视图建模；如果使用星型模式的数据模型做报表和计算更适合应用，那就直接使用分析视图建模；如果计算比较复杂，分析视图满足不了需求，还可以使用计算视图。当上述图形化方式都不能满足需求时，我们可以是用 SQL Script 通过代码创建存储过程来处理复杂计算。如果大家对第八章的决策树的概念还有印象的话，应该记得 SQL Script 编写的计算视图或者存储过程是处于树的最末层的(见图 10-42)。

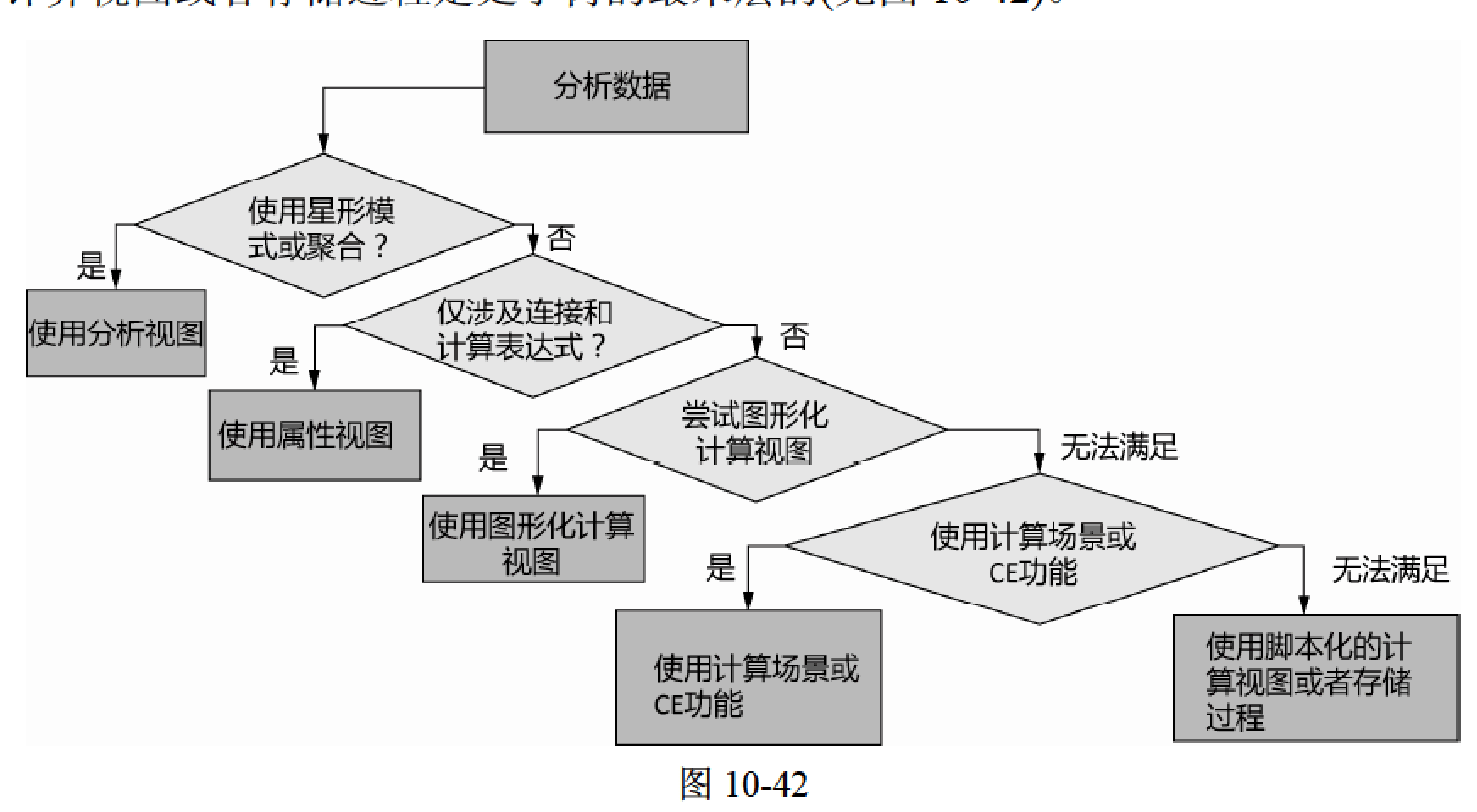


图 10-42

我们之所以在决策树的最末端放置 SQL Script 脚本，原因在于 SAP HANA 的各个引擎分别对属性视图、分析视图和计算视图做过优化，使得各视图能被快速有效地执行；其次 SQL Script 虽然能在数据库层面上进行优化，但是如果其包含控制语



句，则其优化的效果会非常不好。究其原因是因为控制语句的代码对 SAP HANA 来说好像就是一个黑盒，相应的代码基本在 SAP HANA 中会按顺序照原样执行，没有太多的优化和并行运算可以操作。但如果我们可以把控制语句的逻辑用 SQL 直接表示出来，执行的速度会快很多。毕竟，数据库的强项是使用声明语言，生成执行计划，再做优化和执行。

所以，有大多数情况下，我们尽量避免在 SQL Script 中使用不必要的逻辑控制语句，只有在必须使用时才使用它。当逻辑比较复杂时，我们需要尽量做到模块化，将复杂语句分解成小的易于理解和管理的语句，避免复杂难懂的语句，不仅让其他开发人员理解语句，而且要让 SAP HANA 系统也可以清晰快速理解和判断代码的逻辑。

二、避免过于复杂的 SQL Script

写过程序的人可能都知道，为了有效管理程序的复杂性，我们会使用“分解组建/模块化”技术，比如功能模块“A”调用功能模块“B”、ABAP 的 Class、Class 上面的 Method、Sub Method 调用等。用这样技术，我们可以把应用程序分拆成小的易于管理的组件。同时对于通用的逻辑，我们可以写出可以重用的组件作为通用的服务组件，避免相同逻辑重复编写代码。对于 SQL Script，就像上面提到的，这样的想法基本也适用，通常顶层是应用入口，可能仅包含编配(orchestration)的逻辑，即对其他子执行计划的调用。具体逻辑通常在子执行计划中实现。



三、充分考虑数据流图

在模块化这一点上，对于 SAP HANA SQL Script，我们在设计应用的框架时同时应该注意 SQL Script 是面向数据流图的(data flow graph)一种脚本语言，是“Code to Data”。如果很多计算是针对同一个数据集合，我们需要尽量做到在同一个引擎或模块中对这些处理数据计算一次，然后把结果传递到其他的组件，特别是需要把数据在不同的引擎间传递的情况下我们更要注意这一点。所以，如果你设计了一个 SAP HANA 应用，其中包含了很多的存储过程，但都使用同一个数据作为“Input”，并且还频繁地在不同引擎间传递数据，那么你需要仔细考虑是不是设计合理了，比如存储过程分解重用的粒度是否过于细小。

四、小心使用基于行的运算

除了要对上述的数据流图方面进行考虑，在使用 SAP 提供的标准函数时同样也需要留心。大家都知道 SAP HANA 数据库中数据缺省是基于列存储的，所以在 SAP HANA 中对数据做列相关的运算会很快，比如像 SUM、MAX、AVG 这样的聚合运算，但是列存储的数据做行相关的运算就不一定很有效率了。所以应尽量避免过多的基于行的运算。比如我们做了很多的列运算，突然需要做一个行的运算，例如 LOG(b，n)或者 greatest(a，b)，这意味着 SAP HANA 需要传输中间的计算结果，即把数据从列引擎(Column Engine)传递到行引擎(Row Engine)，包括数据的转换和缺失操作符的补充等。等行引擎计算完毕，又需要再次转换，把数据传递回到列引擎，这样在不同引擎间的传递会大大降低执行效率。也许不用行运算，你可以找到其他或类似的列运算[LN(a)]来解决相同的问题。简而言之，尽量避免对列存储数据的行运算还有不必要的在不同引擎间的数据传递。

五、理解 SQL 语句成本

虽然 SAP HANA 是基于内存的计算平台，有各种引擎来优化处理相应的数据操作，但是我们也必须知道每个数据操作都有它相应的成本，有些效率高，有些则比较耗时。

“SELECT(single)*from table”在 ABAP 中很多人使用，但在 SAP HANA 中，这个语句执行效率非常低，这是因为 SAP HANA 中的数据表缺省是列存储的，为了读取“*”，即所有条目，SAP HANA 需要一一读遍数据表中所有的列。

再举个例子，比如仅做数据和合并，UNION ALL 就比 UNION 操作操作效率高，因为 UNION 需要将重复的数据清除。而做清除重复数据前，为了发现这些重复数据，需要对数据做排列或者散列，再形成结果，这样多出来的操作比仅仅使用 UNION 要耗时许多。所以在写 SQL Script 时需要知道相应语句的效果和相应的成本。

你可以从视图“SYS.QUERY_PLANS”查看具体的“Query Plan”信息来帮助你具体了解查询的成本。另外也可以查看 “Performance”分页下的“SQL Plan Cache”数据(如图 10-43 所示)。



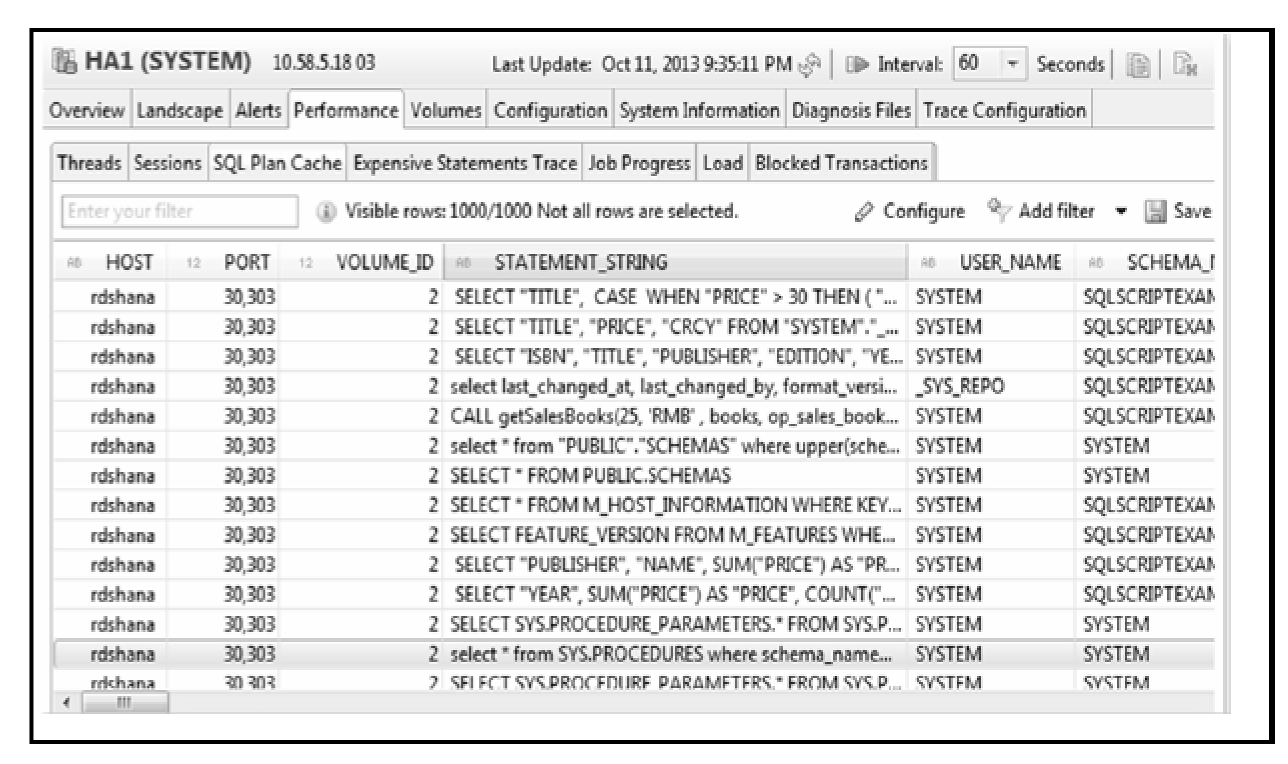


图 10-43

或者打开“Expensive Statements Trace”跟踪是否有特别耗时的 SQL 语句。图 10-44 为设定阀值的界面。

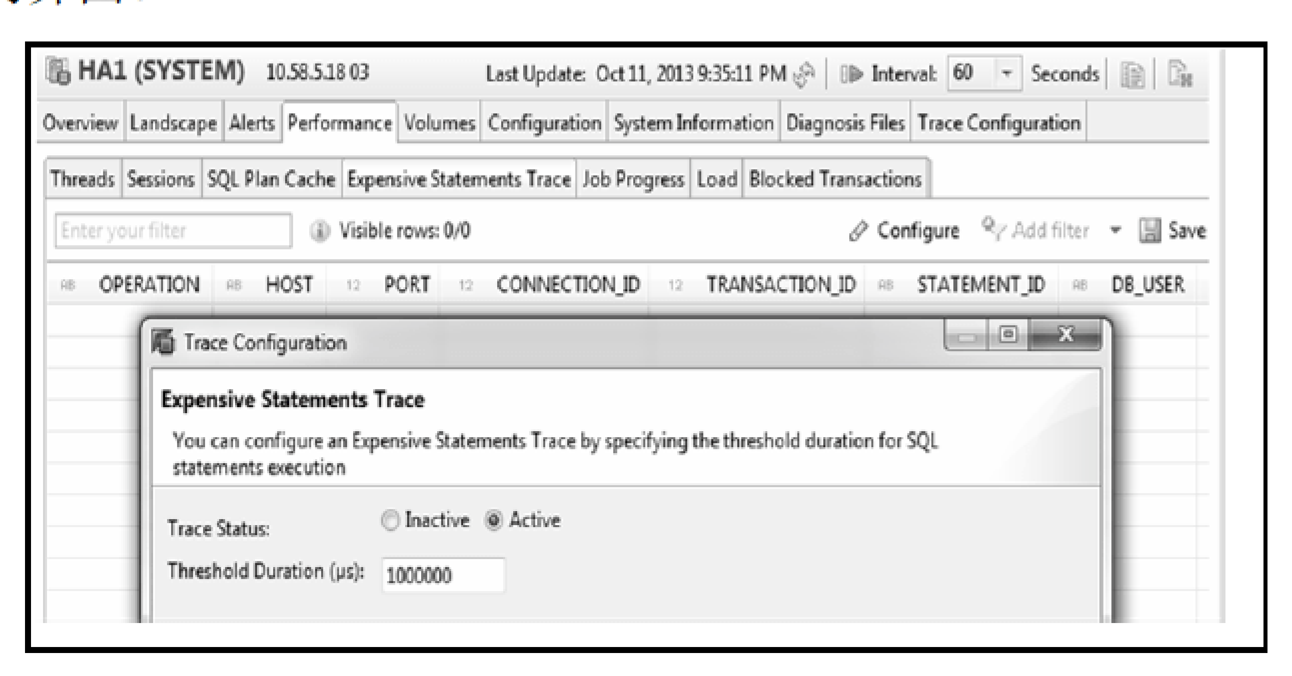


图 10-44

从 SPS04 开始，SAP HANA 还提供了图形化的查看方式——虚拟视图(Visualize Plan)。比如当我们写完某个存储过程或者复杂的 SQL 语句后，如果想查看 HANA 到底是如何执行，每步花了多长时间，我们可以查看虚拟视图。如图 10-45 所示，右击“call procedure”的 SQL 语句，选择“Visualize Plan”。

然后选择“Execute”(见图 10-46)。

虚拟视图会给出执行的时间数据，单击右侧的三角，你还可以进一步下钻到更细节的执行数据(见图 10-47)。



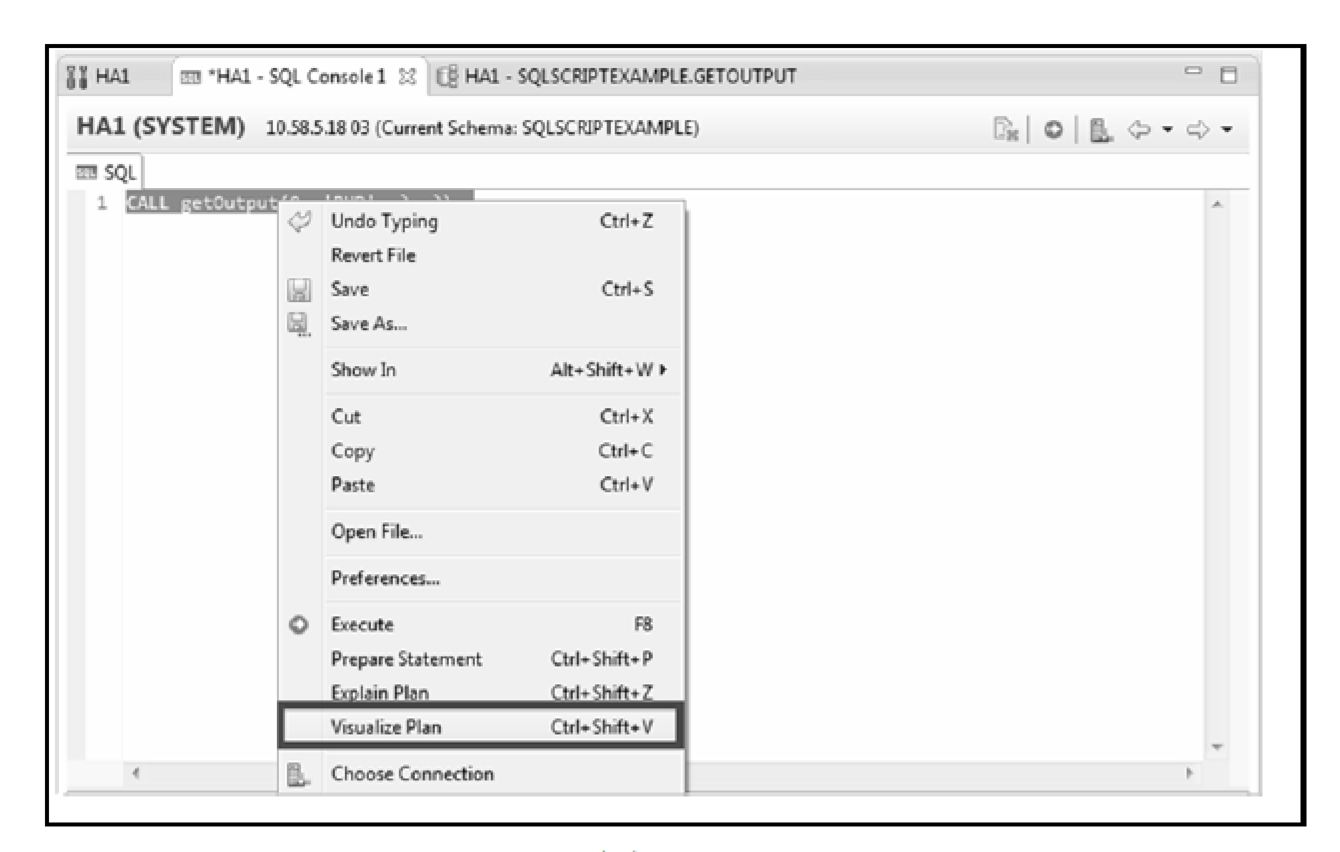


图 10-45



图 10-46

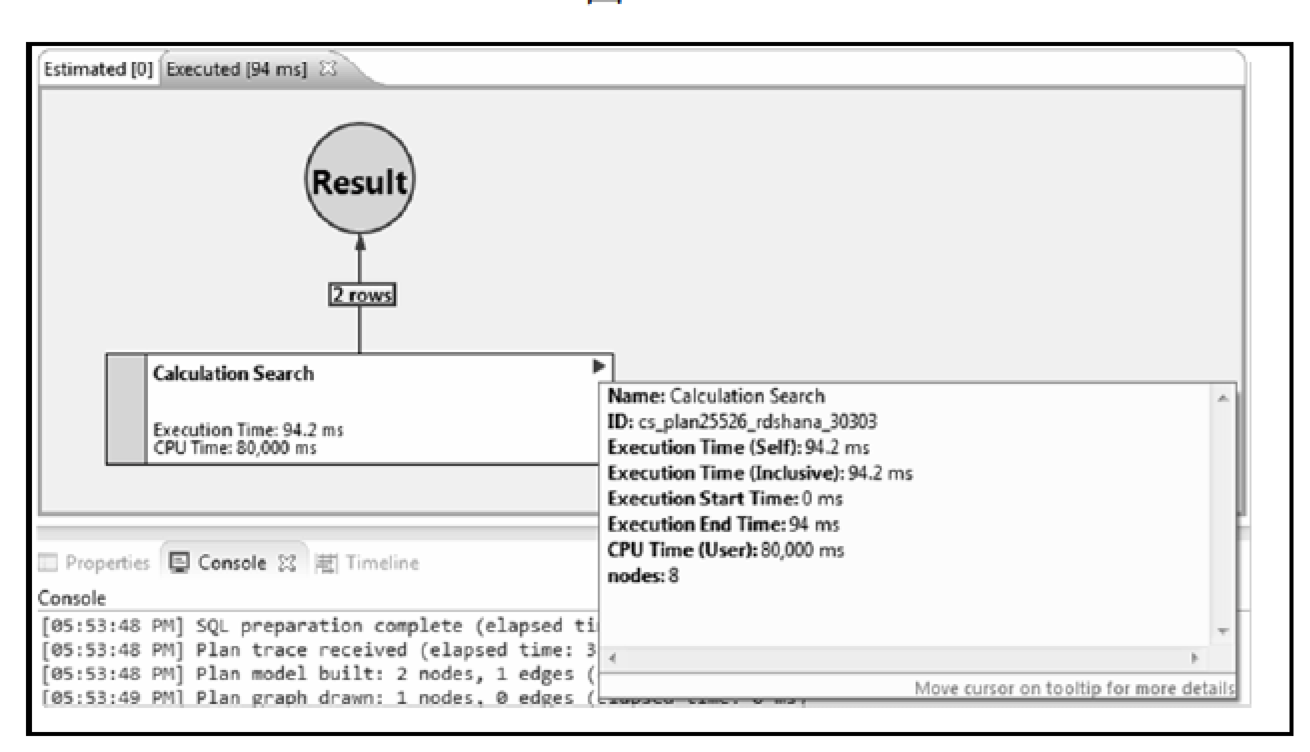


图 10-47

下钻后看到如图 10-48 所示的数据。



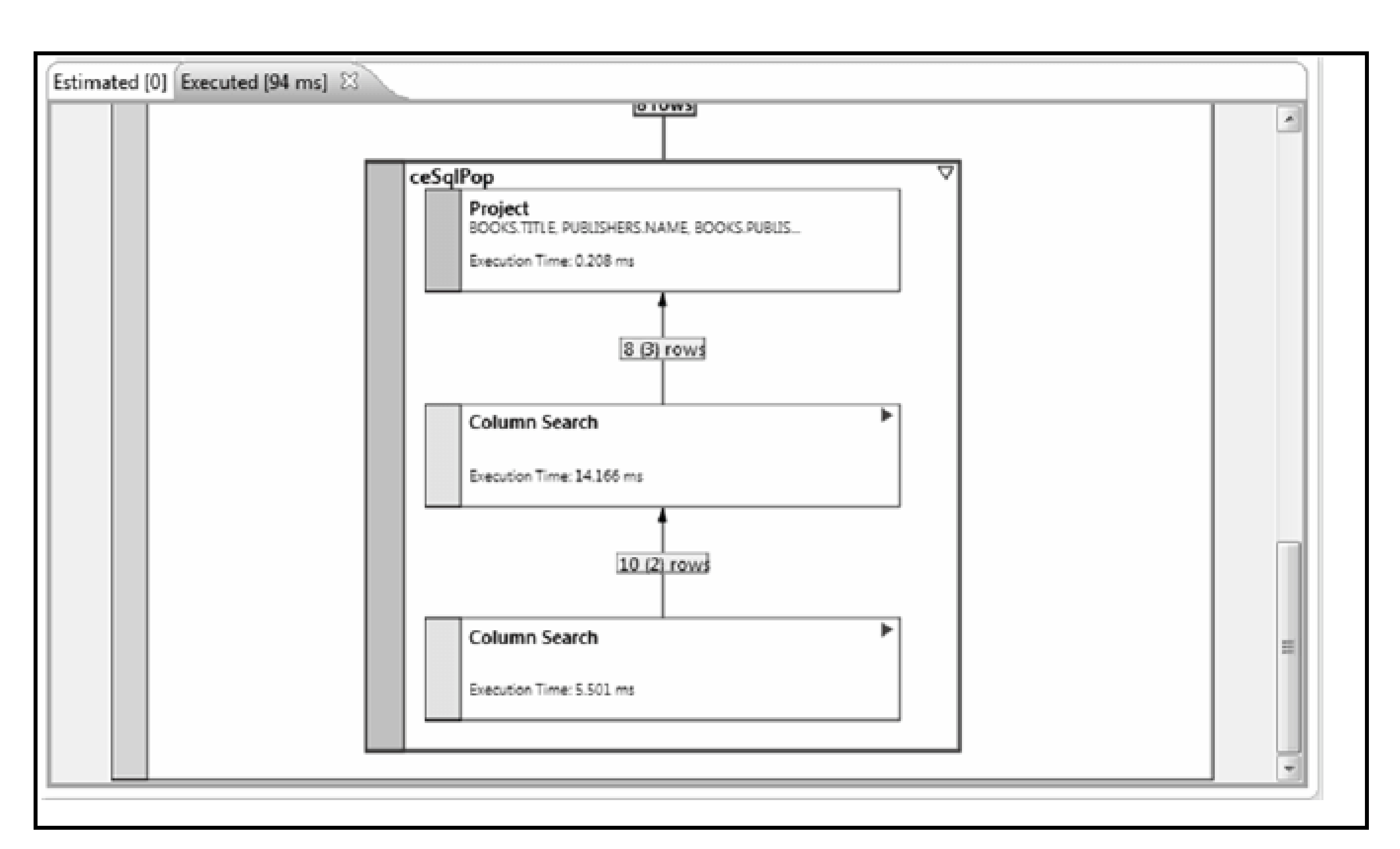


图 10-48

进一步下钻展开的结果如图 10-49 所示。

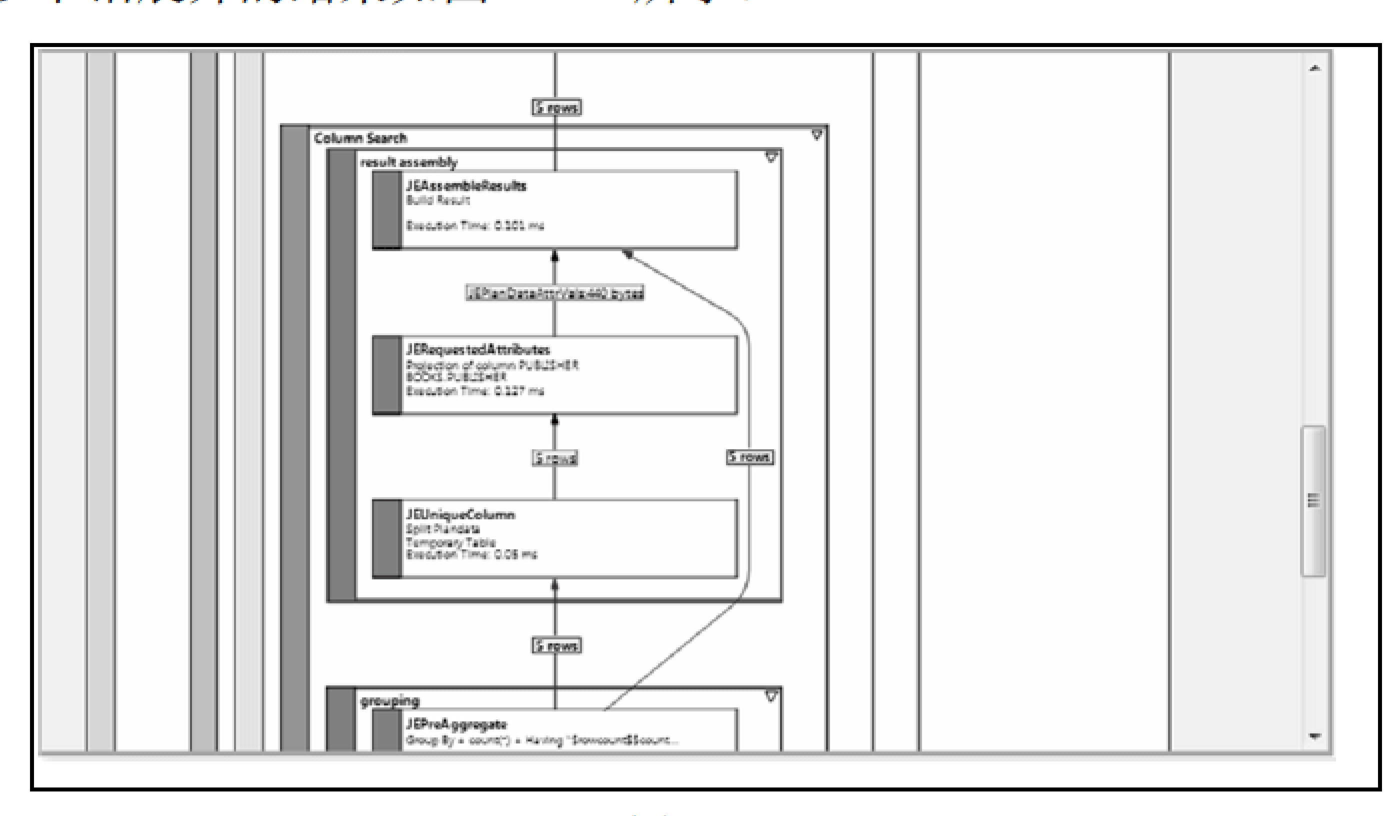


图 10-49



六、只读存储过程与可读写存储过程

大家前面可能已经看到，在定义存储过程时，我们可以定义存储过程是“read-only”(只读)或者是“read-write”(可读写)类型。需要注意的是，对于可读写的存储过程，SAP HANA 是按顺序依次执行的。而且，如果当前的存储过程定义为可读写类型，调用此存储过程的 call stack 上层的存储过程也必须是可读写，且会依次按顺序执行。如图 10-50 所示。

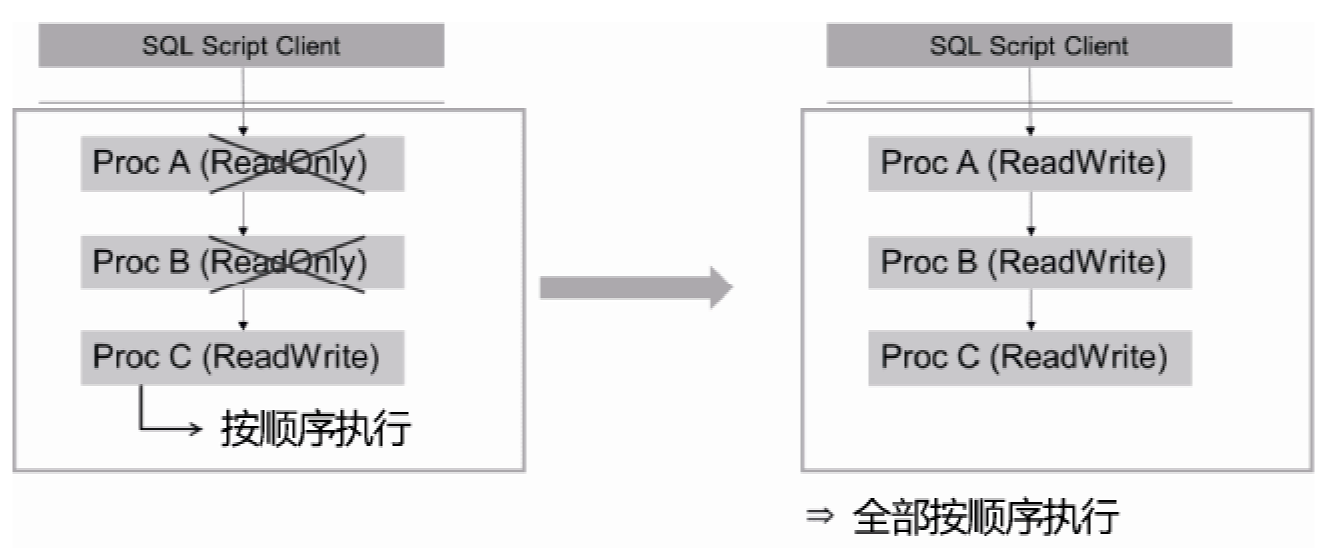


图 10-50

七、避免数据依赖性

大家在前面 SAP HANA 介绍的章节已经了解 SAP HANA 运行高速一个重要原因就是支持并行执行，并且其可以在不同的层次和粒度。举个例子，来自不同用户的不同的数据处理请求可以被并行执行，某个用户的单个查询请求中的关系运算也可以在服务器上的多个核上并行执行。一个存储过程中的不同的独立的语句也可以并行运行。

就如前面提到的，SQL Script 的逻辑会被转换到数据流图，我们必须注意到，只有独立的路径才可以被并行执行。简单地说，并行的前提是数据的独立性，这很好理解。我们在写 SQL Script 的时候，需要注意在不同计算逻辑间要避免不必要的数据依赖，并且尽量使用 Declarative 语句。比如尽量避免变量引用和游标的滥用。

八、避免滥用游标

写惯了 ABAP 的读者在刚开始接触 SQL Script 的时候可能会问，怎么对内部表的数据做 LOOP 循环，并以此对每个数据进行诸如比较大小、加总计算等循环类的操作。SQL Script 提供类似这样一种 LOOP 循环的机制，通过使用游标来完成。但请注意，在我们使用游标的时候，也就意味着我们在执行基于行的操作，这样的操作应尽量避免，除非没有其他基于列的处理方式。因为 SAP HANA 数据库中数据默认都是以列的形式存储的，SQL 引擎对这样的行操作优化的空间较少。所以在你很习惯性地做 LOOP 操作时，请仔细考虑是否有其他的方法，以及这样的操作是不是必要。

下面是初次接触 SQL Script 的人有可能会犯的低级错误，其实是直接可以通过列操作完成。这里仅做个例子，给大家提醒。



下面这个存储过程通过对游标进行循环来进行价格的求和运算：

```
CREATE PROCEDURE badforeach_proc LANGUAGE SQLSCRIPT AS
  val decimal(34,10) := 0;
CURSOR c_cursor1 FOR SELECT isbn, title, price FROM books;
BEGIN FOR r1 AS c_cursor1 DO
val := :val + r1.price;
END FOR;
END;
```

我们直接通过基于列的汇总一样能完成同样的价格总和计算：

```
SELECT sum(price) into val FROM books;
```

九、避免动态 SQL

动态 SQL(Dynamic SQL)是一个非常强有力的工具，特别是在表达程序逻辑时。它允许在存储过程运行的时候根据需要来动态构造 SQL 语句。但是，执行动态的 SQL 也会较慢，因为在每次动态语句被调用时，都需要做相应的编译检查和查询优化。所以，只有在真的没有其他办法而需要动态地构建 SQL 语句的情况下，才去使用这样的工具，切勿随意滥用。

十、SAP ECC 客户端处理

通常，基于 ABAP 的应用程序和数据表都有客户端(Client)这样的概念，当我们把 ECC 的数据同步到 SAP HANA 后，在建模或者写存储过程时需要做相应的处理。

我们先看建模的情况。以分析视图为例，在 SAP HANA Studio 里面打开某个分析视图，查看属性，如图 10-51 所示，我们可以看到“Default Client”这个属性。

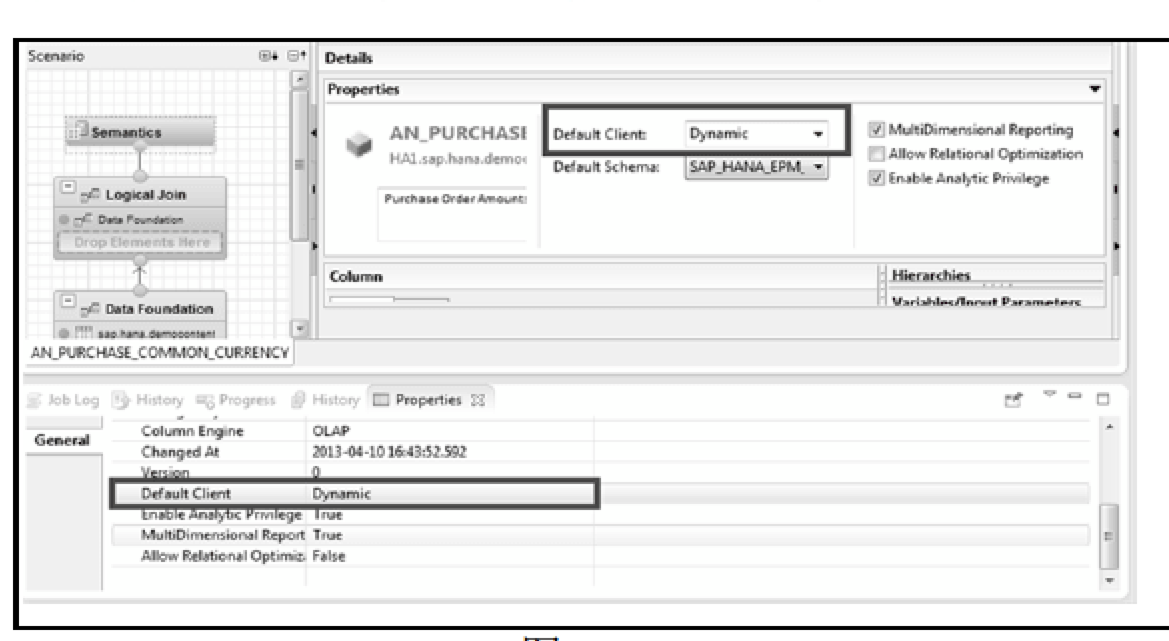


图 10-51

我们可以根据需要选择设定为“Dynamic”，这样系统会根据用户的客户端来过滤数据。用户的客户端可以在用户页面设定(见图 10-52)。

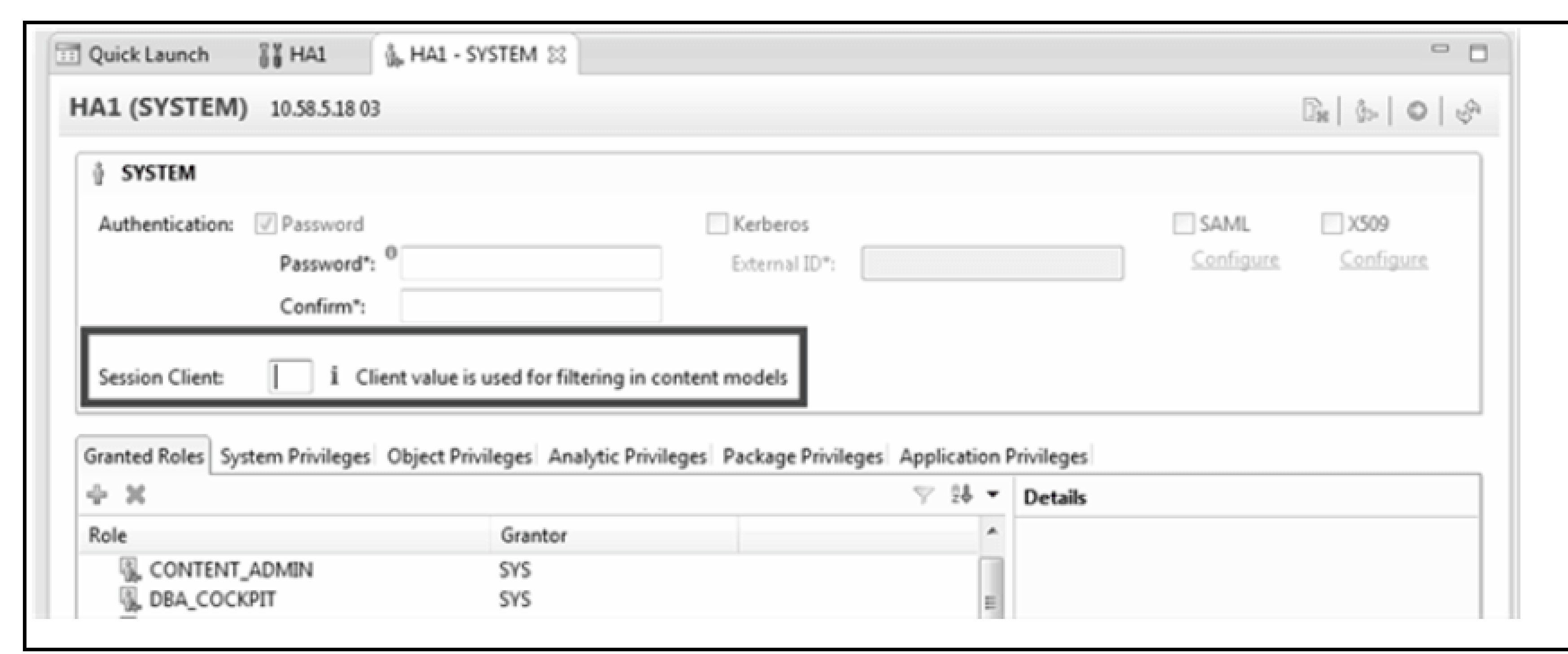


图 10-52

对于计算视图，如果我们需要动态的选择数据，可以使用 SESSION_CONTEXT('CLIENT')，例子如下：

```
SELECT field1, field2 FROM « my_table » WHERE mandt = SESSION_CONTEXT('CLIENT')
AND spras = SESSION_CONTEXT('LOCALE_SAP')
```

除了上面两种方式，在实际操作时，如果不想每次查询的时候都带上 MANDT 参数，我们也可以通过自己手动创建视图的方式去实现。

```
CREATE COLUMN TABLE "DEMO_TABLE_MANDT"(
    "MANDT" NVARCHAR(3),
    "FIELD_ID" BIGINT,
    "FIELD_TYPE" NVARCHAR(10),
    PRIMARY KEY ("MANDT", "FIELD_ID")
);

CREATE VIEW "V_DEMO_VIEW" ( "FIELD_ID", "FIELD_TYPE" ) AS
SELECT "FIELD_ID" AS FIELD_ID,
       "FIELD_ID" AS FIELD_TYPE
FROM "DEMO_TABLE"
WHERE "MANDT" = SESSION_CONTEXT('client');
```

上语句中的“SESSION_CONTEXT”中的“client”只需在前面做一次设置操作即可。当视图生成后，以后的数据的读取写入都针对此视图操作即可。



第五节　通过 ABAP 调用存储过程

一、基础简介

在介绍 ABAP 调用 SAP HANA 存储过程前，我们先了解下 SAP HANA ABAP 应用的两种基本架构。一种是把 SAP HANA 作为附属数据库连接，通过 ELT 工具同步 ECC 和 SAP HANA 上的数据，新应用在 SAP HANA 上运行，其主要用途是为了业务流程加速。另外一种是把 SAP HANA 作为主数据库，全部应用的读写操作都在 SAP HANA 数据库上进行。图 10-53 给了图形化的解释，左边图是把 SAP HANA 作为附属数据库用做加速器，右边 SAP HANA 是作为主数据库来运行全部应用。

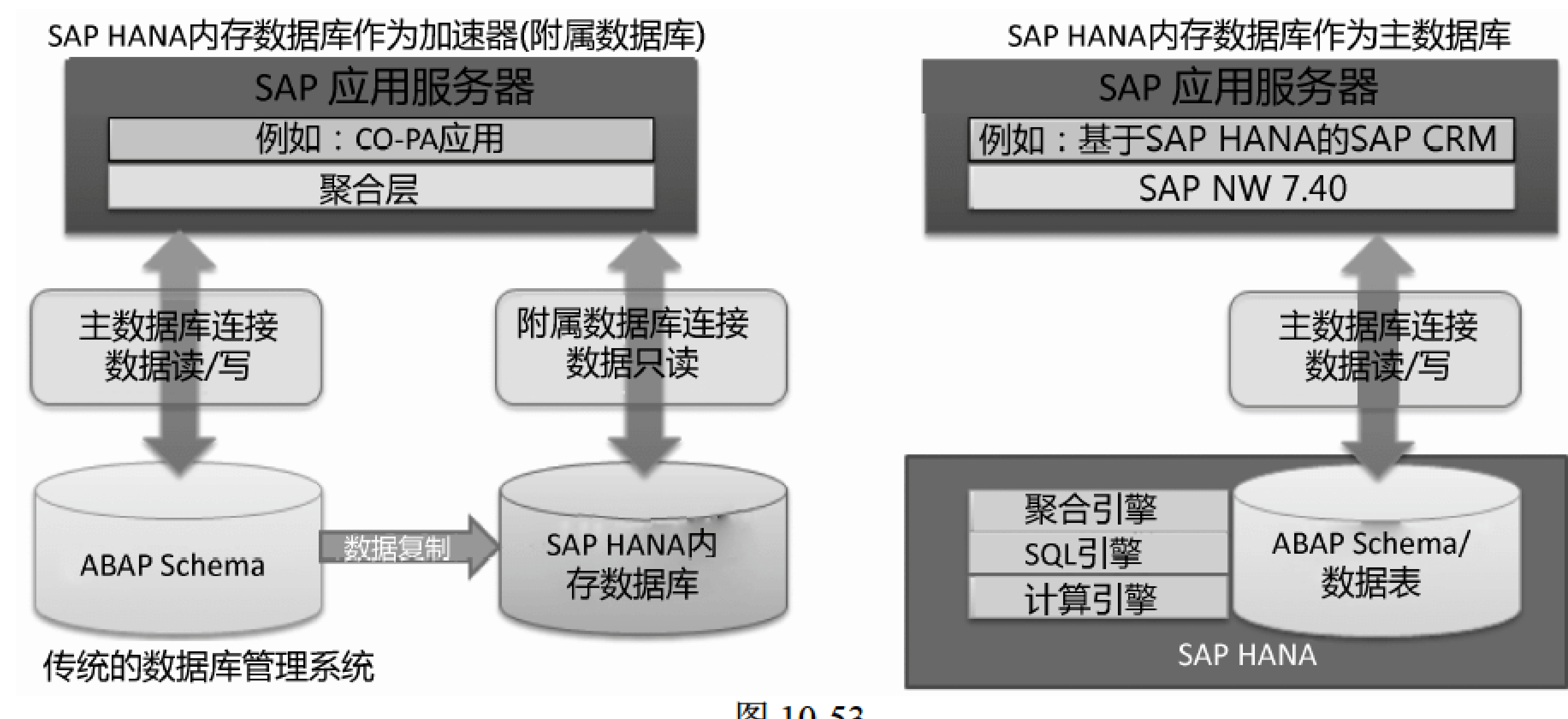


图 10-53

对于上述两种情况，如果 NW ABAP 版本是 7.40 以上，在创建了 Database Procedure Proxy 后，可以在 ABAP 直接使用 CALL DATABASE PROCEDURE 调用 SAP HANA 的存储过程，如同调用普通的 ABAP 功能模块一样，如图 10-54 所示。

我们在编写 ABAP 程序的时候，经常需要从数据表中根据某些条件读取数据，读取数据最常用的方法就是通过 SQL 语法实现的。ABAP 语言可以使用的 SQL 语句有 Open SQL 语句和 Native SQL（既原生的 SQL）语句。Open SQL 语句不是标准的 SQL 语句，是 ABAP 语言，利用 Open SQL 语句能在数据库和命令之间产生一个缓冲区，所以它有一个语言转换的过程。而 Native SQL 语句则是标准的 SQL 语句，它直接针对数据库进行操作。

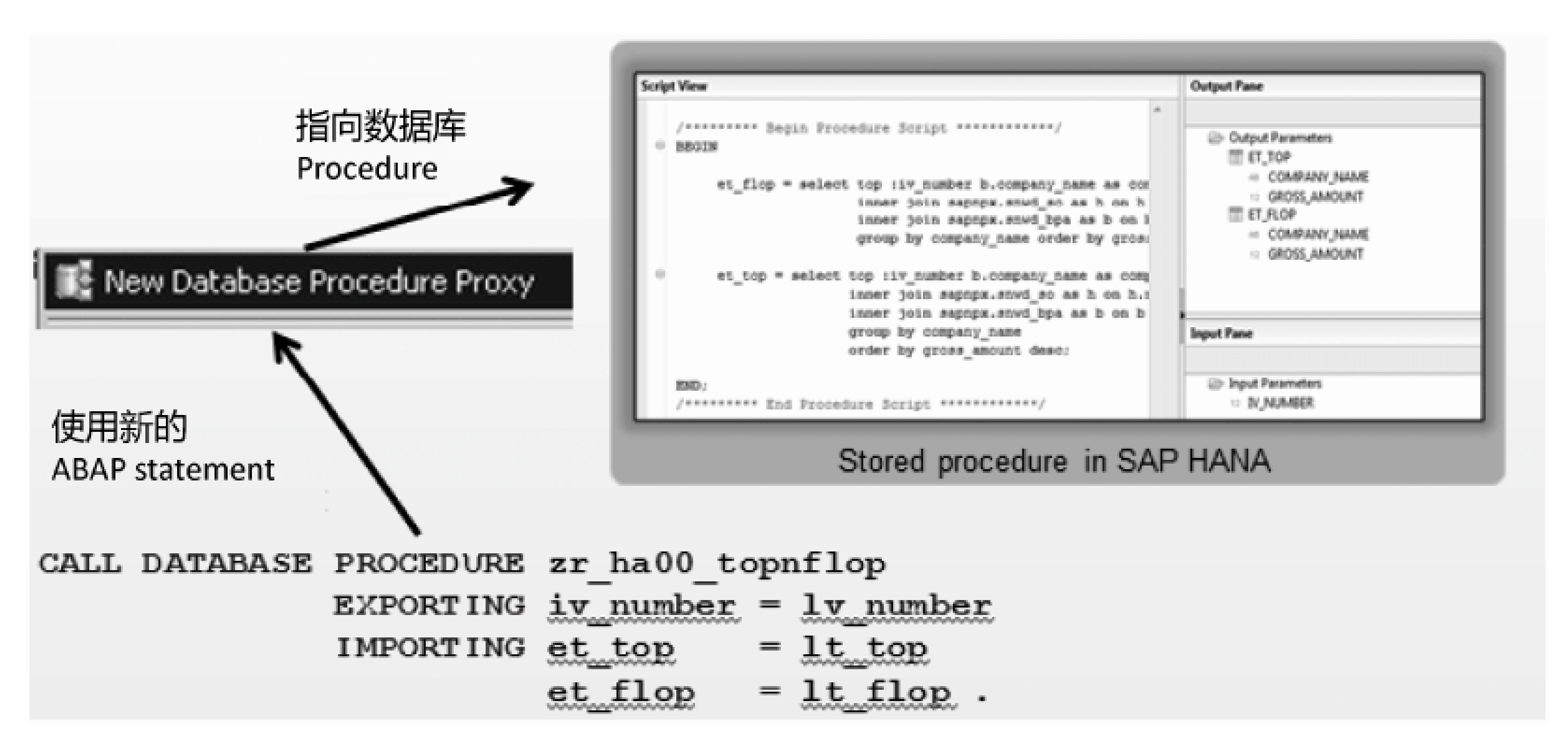


图 10-54

这里也顺便指出，如果 ABAP 版本为 7.40 及以上，SAP HANA 中的数据表和视图也可以在 ABAP 中通过 Open SQL 直接引用。

对于 ABAP 版本在 7.40 以下的情况，我们还是需要通过 Native SQL 调用存储过程。对于 Open SQL 和 Native SQL 概念还不是太清楚的读者，这里我们简单介绍其区别(见图 10-55)。

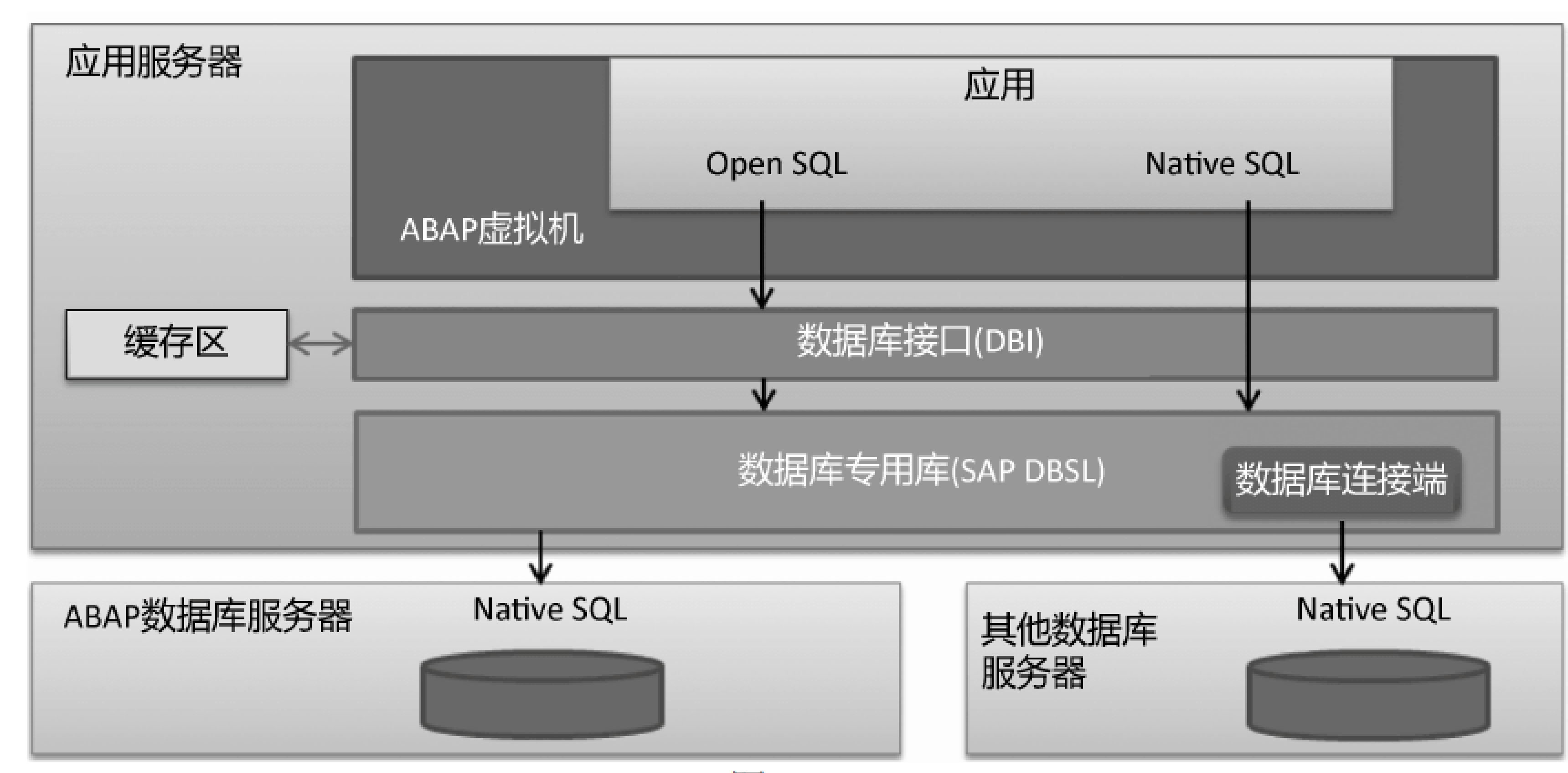


图 10-55

图 10-55 是 ABAP 的数据库架构，通过 DBI 和 DBSL，SAP ABAP 系统可以连接到不同的数据库系统。

在 SAP ABAP 中，我们可以通过 Open SQL 进行数据操纵语言的操作，比如数据的增删改查和相关操作，如图 10-56 所示。数据控制语言则通过“SE11”事务代



码在数据字典(Data Dictionary)中实现。如要在 SAP NW ABAB 7.40 版本前的 SAP NW ABAP 平台上使用 Open SQL，则仅能在数据字典中创建过的表和视图上操作。

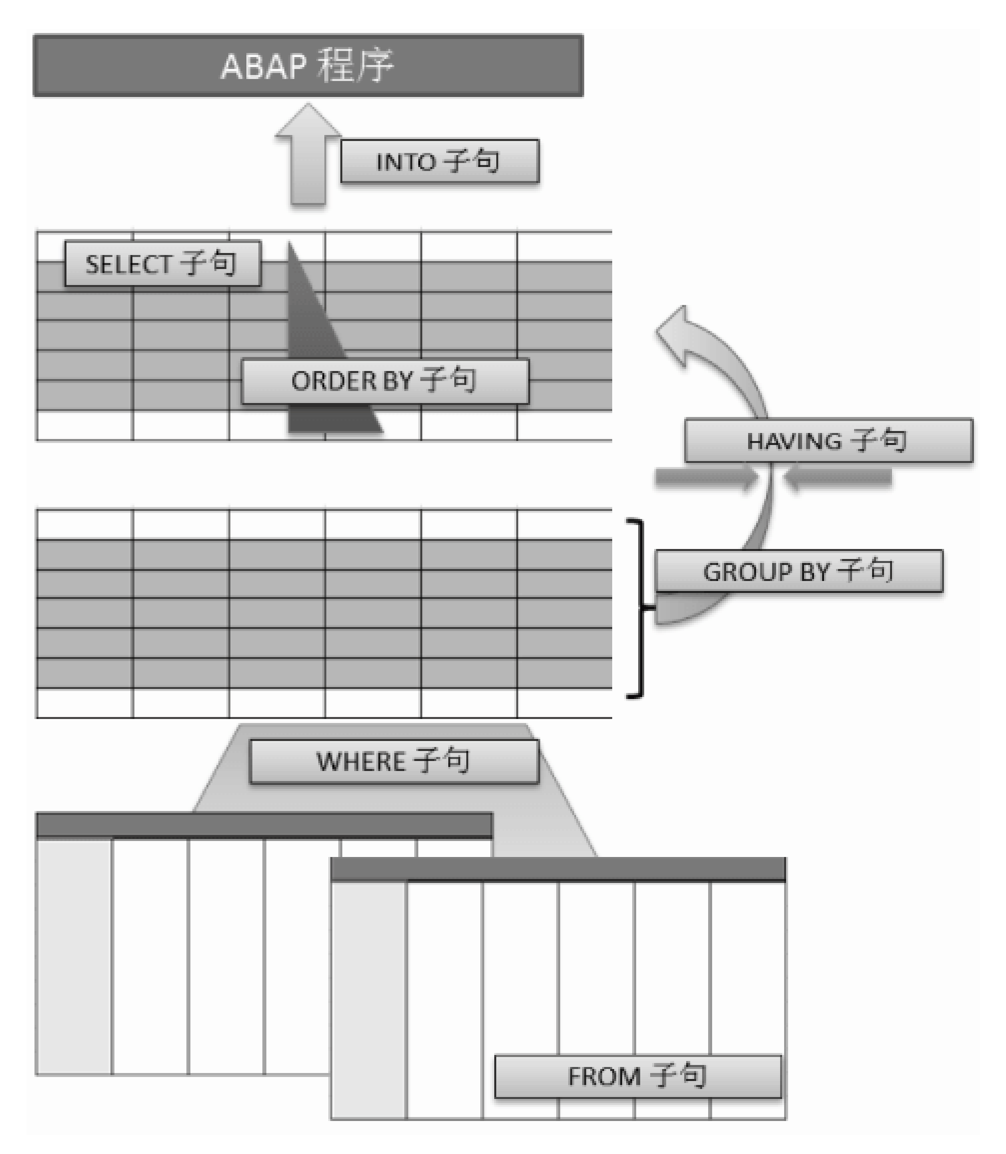


图 10-56

由于 ABAP Open SQL 的局限性，目前我们推荐通过 Native SQL 来灵活读取 SAP HANA 的数据表视图的数据，调用 SAP HANA 中的存储过程。

调用的方式有两种：

- 通过 EXEC SQL、ENDEXEC(如图 10-57 所示，不推荐)
- 基于 ABAP Class 形式的 ABAP Database Connectivity (ADBC)




```
EXEC SQL.
  CONNECT TO 'ACI' AS 'ACICONN'
ENDEXEC.

EXEC SQL.
  create column table "SYSTEM"."BOOKS"
  ( "ISBN" VARCHAR (20), "TITLE" VARCHAR (40),
   "YEAR" VARCHAR (4), "PRICE" DECIMAL (5,2),
   "CRCY" VARCHAR (3))
  WITH PARAMETERS ('SESSION_TYPE'='SIMPLE' )
ENDEXEC.
```

图 10-57

二、建立 ADBC 数据库连接

下面我们介绍从 ABAP 端通过 ADBC 调用存储方式这一方式。其方法也很简单，包含如下步骤：

- 创建数据库的 Secondary Connection，此为一次性操作；
- 创建 CL_SQL_STATEMENT 实例；
- 调用 execute_query()实现 Native DB Cal；
- 通过 set_param() set_param_table()取得结果集变量；
- 通过 next_package()方法取得结果集数据。

下面是创建数据库连接步骤：

(1) 登录 SAP ECC 系统，运行“DBCO”(见图 10-58)。

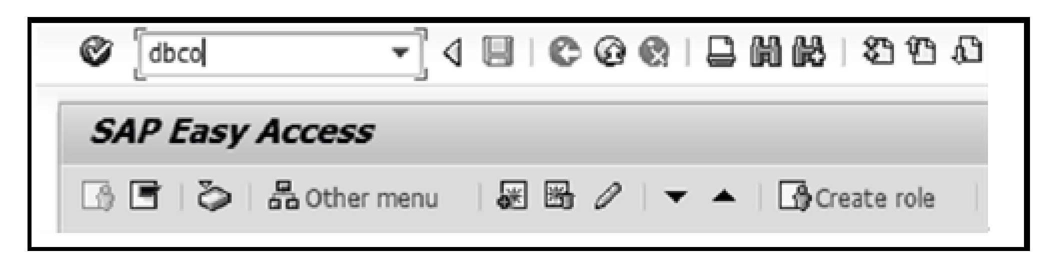
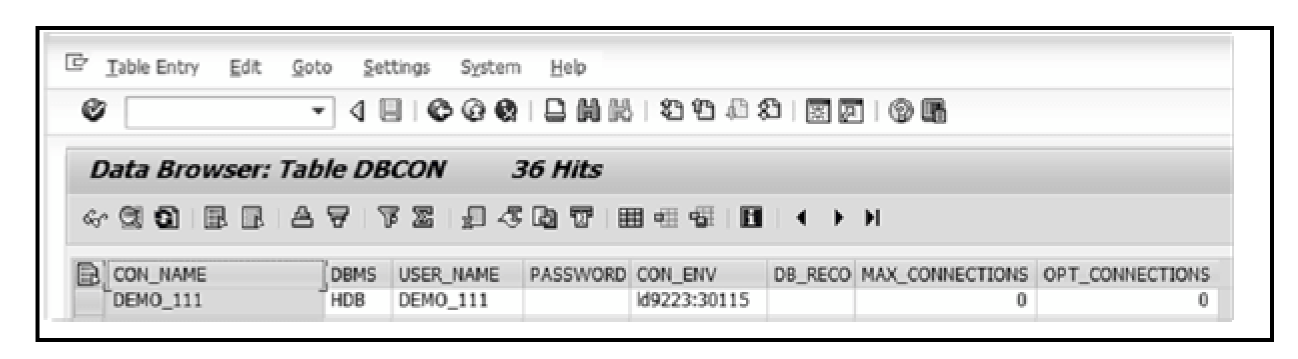


图 10-58

(2) 配置 DB Connection，如图 10-59 所示。除了指定名称外，还需要数据库类型，服务器及其端口、用户名和密码等。



CON_NAME	DBMS	USER_NAME	PASSWORD	CON_ENV	DB_RECO	MAX_CONNECTIONS	OPT_CONNECTIONS
DEMO_111	HDB	DEMO_111		ld9223:30115		0	0

图 10-59

(3) 创建 CL_SQL_STATEMENT 实例。先通过 cl_sql_connection 取得链接“lo_dbconn”，接着创建 SQL Object 的实例“lo_sql”，通过实例的“execute_query”方法我们就可以调用 SQL 语句了(见图 10-60)。

```
lo_dbconn = cl_sql_connection=>get_connection( con_name = 'DEMO_111' ).

CREATE OBJECT lo_sql
  EXPORTING
    con_ref = lo_dbconn.

lo_result = lo_sql->execute_query( 'SELECT * FROM MY_HANA_TABLE"' ).
```

图 10-60

三、调用存储过程

图 10-61 中这段代码给出了把 ABAP 中的变量值传递到 SQL 中的方法。

```
lo_sql->execute_query( 'SET SESSION ''client'' = ''' && sy-mandt && '''' ).

lo_sql->execute_ddl( 'TRUNCATE TABLE "IT_PROD_IDS"' ).
LOOP AT it_prod_ids INTO lv_prod_id_input.
     lo_sql->execute_update( 'INSERT INTO "IT_PROD_IDS" VALUES(''' && lv_prod_id_input &&''' )' ).
ENDLOOP.


lo_sql->execute_ddl( 'TRUNCATE TABLE "OT_COMPOSITION"' ).


lo_result = lo_sql->
execute_query( 'CALL "_SYS_BIC"."sap.teched/EXPLOSION"( "IT_PROD_IDS", "OT_COMPOSITION" ) WITH OVERVIEW ' ).

lo_result->close( ).


lo_result = lo_sql->execute_query( 'SELECT * FROM "OT_COMPOSITION"' ).
```

图 10-61

本章小结与练习

本章我们从 SAP HANA 的创新理念，将计算逻辑从应用层下放到基于内存的数据库层开始，了解了为什么从传统数据到代码方式转换到代码的数据可以大幅提高大数据运算效率。为了使计算在数据库层成为可能，SAP HANA Script 在如下三个方面对标准 SQL 进行了扩展，来支持这种在数据库层计算的理念。

- 数据类型扩展：容许定义“Table Type”，即使没有相应的数据库基础表。
- 函数扩展：容许处理复杂数据流，比如支持多个输入输出、存储过程嵌套。
- 规则扩展：容许规则程序的执行，比如数据更新。IF ELSE、LOOP 等规则逻辑控制结构。

我们了解了相对标准 SQL，SQL Script 有了如下优势：

- 一个 SQL Query 仅能返回一个数据集，而存储过程可以返回多个数据集。
- 对于复杂的查询，SQL 仅能使用 SQL 视图来构造实现，但视图没有相应输入输出参数可以使用；而使用 SQL Script，我们可以定义输入输出，进行模块化，拆分复杂的函数逻辑，将复杂逻辑细分成子逻辑，增强程序的结构性、可读性和重用性。



- SQL Script 支持对中间步骤数据定义临时本地变量，而 SQL 需要定义全局视图，即使是为中间临时数据使用。
- SQL Script 支持规则控制，比如 IF/ELSE、LOOP 等。

为了更好理解 SQL Script 的语法和优化基础，在讲解基本 SQL Script 语法前，我们还了解了存储过程在 SAP HANA 系统中的创建、编译和执行过程，以及数据流图在优化中的重要角色。而业务流程逻辑和声明逻辑也是编译执行优化需要理解的基本概念。

- 声明逻辑可以简单地理解为 SQL SELECT 和 CE Function 语句，负责读和计算语句的高效执行，可以多层次，多粒度的并行执行；
- 业务流程逻辑通过 DDL、DML、查询语句和规则语言结构(比如 LOOP 和条件语句)来实现和控制数据流，负责协调组织。规则逻辑(比如 IF ELSE 循环控制等)一般按顺序依次执行。

了解完这些概念，我们通过 SAP HANA Script 中的标准帮助文档中的“PUBLISHERS & BOOKS”的例子一一讲解了创建存储过程的基本语法，包括创建存储过程需要的基本表和数据、输入输出需要的数据表类型定义、定义多个输入输出、简单只读存储过程程序构造、存储过程的调用(通过 SAP HANA Studio 客户端、其他存储过程、SAP ABAP 客户端调用)、WITH OVERVIEW 参数(将输出数据写入到表中)、WITH RESULT VIEW 参数(创建一个新的视图并插入相应输出数据)。

到这一步，我们可以理解为 SQL Script 的函数方面的扩展。

了解函数方面的扩展和存储过程创建的基本语法后，我们更近一步学习了 SQL Script 规则扩展相关内容，即如下主要技术要点：

- 如何通过“:”引用输入变量和本地变量，如何为临时本地变量赋值(:=)；
- 如何通过 IF ELSE、WHILE 循环、FOR 循环控制程序结构；
- 如何使用 CURSOR(open，close，fetch，row 读取)；
- 如何使用动态 SQL；
- 最后讲解了 CE FUNCTION(data access & relation)的使用。

这一部分可以理解为 SQL Script 的规则扩展。

之后，我们大致了解了 SAP HANA Script 的开发注意事项，也可以叫做最佳实践吧，其中包括：

- 充分理解和使用 SAP HANA 内置引擎(基于数据特性分析需要用那种视图建模，是否需要写存储过程来满足复杂计算)；

- 避免复杂的 SQL Script(通过模块化)；
- 小心使用行存储引擎(避免大数据在不同引擎间的传递)；
- 理解 SQL 语句成本(以“select*”语句为例，大量的这类语句会大大增加处理时间，你可以使用 SAP HANA 自带的性能相关工具来检测所编写 SQL 语句的执行效率)；
- 只读和读写存储过程(读写存储过程的上层也必须为读写存储过程)；
- 避免数据依赖性(以实现并行运行)；
- 避免滥用指针(Cursors)和动态 SQL(指针为行操作，动态 SQL 执行需重新编译)；
- 客户端处理的小技巧；
- 已经在 ABAP 中通过 ADBC 调用 SAP HANA 的存储过程。

练习

1：EPM

第一个练习较为简单，直接把你在上一章针对 EPM 的 SQL 语句为基础做下改进，嵌入存储过程即可。

我们定义存储过程为 READ ONLY，请计算供应商 A 的产品 B 和 C 在某年的所有 PO 的价格总和。

输入有两个参数：供应商 ID，ProductID(table，多个)。

输出为：PO 价格总和，GROUP BY 年度。

请自己尝试实现这个存储过程并在 SQL CONSOLE 中调用查看数据。

2：推荐算法 Slope one

问题描述：

Slope one 是一系列应用于协同过滤的算法的统称。协同过滤算法是基于一个个体过去对某些项目的评分和(大量)由其他用户的评价构成的数据库，来预测该用户对未评价项目的评分或喜好。

比如，如果一个人给甲壳虫乐队的评分为 5(总分为 5)的话，我们能否预测他对电台司令乐队新专辑的评分呢？

下面举个 Slope one 算法实际的例子，如表 10-2 所示。

表 10-2

用　　户	对事物 A 打分	对事物 B 打分
X	3	4
Y	2	4
Z	4	?

Slope one 算法也认为：平均值可以代替某两个未知个体之间的打分差异。

我们根据已有数据知道，事物 A 对事物 B 的平均差值是：[(3−4)+(2−4)]/2=−1.5，也就是说人们对事物 B 的打分一般比事物 A 的打分要高 1.5。

Slope one 算法就猜测 Z 对事物 B 的打分是 4+1.5=5.5。

再举个稍复杂，带有权重的例子。我们想预测顾客珍妮对项目 1 的评分。

表 10-3

顾客	项目 1	项目 2	项目 3
约翰	5	3	2
马克	3	4	未评分
珍妮	未评分	2	5

在本例中，项目 1 和项目 2 之间的平均评分差值为[2+(−1)]/2=0.5，项目 1 的评分平均比项目 2 高 0.5。

项目 3 和 1 之间的平均评分差值为 3。平均评分差值计算公式如下：

$$D_{mj}=\frac{\sum m:user\ mrated\ item\ iand\ j\left(R_{mj}-R_{mj}\right)}{f_{ij}}$$

R_{mj} 代表用户 m 给商品 j 的评分；R_{mi} 代表用户 m 给商品 i 的评分；差异为 R_{mj}−R_{mi}，平均评分差值 D_{ij}；f_{ij} frequency 是同时评论了商品 i 和商品 j 的用户数。

D21=[(5−3) + (3−4)]/2=0.5

D31=(5−2)/1 = 3

如果我们试图根据珍妮对项目 2 的评分 D21 来预测她对项目 1 的评分的时候，我们可以得到 2+0.5=2.5。

如果我们想要根据她对项目 3 的评分来预测 D31 她对项目 1 的评分的话，我们得到 5+3=8。

如果其他用户已经评价了一些项目，可以把各个已经评论过的项目作为权重。在上面的例子中，项目 1 和项目 2 都评价了的用户数为 2(D21)，项目 1 和项目 3 都

评价了的用户数为 1(D31)，因此权重分别为 2 和 1。

Slope one 算法预测珍妮对项目 1 的评分为：

$$\frac{2\times2.5+1\times8}{2+1}=\frac{13}{3}=4.33$$

相应的计算公式为：

$$R_{mj}=\frac{\sum i:user\ mrated\ item\ i\left(R_{mi}+D_{ij}\right)*f_{ij}}{\sum i:user\ mrated\ item\ i\ f_{ij}}$$

简要了解了 Slope one 算法，我们来看看如果通过存储过程来实现和优化。

现在有两个表(“TRAINING”和“TEST”)有数据，output 为输出表，在同一个 Schema 下。

表有相同的字段(UESRID，ITEMID，RATING)，如图 10-62 所示。

Table Name: TRAINING_TAB　Schema: CONTEST　Type: Column Store

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null	Default	Comment
1	USERID	VARCHAR	100	STRING	X(1)	X		
2	ITEMID	VARCHAR	100	STRING	X(2)	X		
3	RATING	DOUBLE		DOUBLE				

图 10-62

上面看到，Slope one 算法有两步，先计算 D_{ij}，再计算 R_{mj}。

第一步：计算平均评分差 D_{ij}，数据存在“TRAINING”表(见表 10-4)。

第二步：计算 R_{mj}，R_{mj}代表用户 m 对商品 J 的评论，数据在“TEST”表(见表 10-5)。

对“TEST”表中的每个用户 m，他们评论了“TRAINING”表中的物品但是还未评论“TEST”表中的该物品。对于缺失的用户 m 对物品 j 的评论，我们通过 Slope one 计算。

用户 2 评论了物品 1 和物品 2，但是未评论物品 3。

用户 3 评论了物品 3，但是未评论物品 1 和物品 2。

缺失的评论数据计算后放入表 RESULT_TAB(见表 10-6)。

TRAINING_TAB：

表 10-4

USERID	ITEMID	RATING
1	1	5
1	2	3
1	3	2

TEST_TAB：

表 10-5

USERID	ITEMID	RATING
2	1	3
2	2	4
3	3	5

RESULT_TAB：

表 10-6

USERID	ITEMID	RATING
2	3	1.5
3	1	8
3	2	6

请尝试建立需要的表，并用存储过程实现。

下面给出一个运行较慢的存储过程例子，你可以尝试运行，再试试可否改进。

```
CREATE PROCEDURE SLOPE_ONE LANGUAGE SQLSCRIPT AS
BEGIN
    result = select t.USERID, fd.itemid1 as ITEMID,
    sum(freq*(diff + rating)) /sum(freq) as rating
            from (
            select ud1."ITEMID" as itemid1, ud2."ITEMID"  as itemid2,
count(*) as freq,
                    (sum(ud1."RATING" - ud2."RATING"))/count(*) as diff
                 from contest."TRAINING_TAB" ud1 join contest."TRAINING_
TAB"  ud2
                 on ud1."USERID"  = ud2."USERID" and ud1."ITEMID" !=
ud2."ITEMID"
                 group by ud1."ITEMID", ud2."ITEMID"
                ) fd
                JOIN
                (
            select * from  contest."TEST_TAB"
                ) t
                on fd.itemid2 = t."ITEMID"
            where  not exists
                (
                 select * from contest."TEST_TAB"
               where t.USERID = "TEST_TAB".USERID AND "TEST_TAB".ITEMID =
fd.itemid1
```




```
        )
    group by t.USERID, fd.itemid1;

    INSERT INTO RESULT_TAB SELECT * FROM :result;
END;
```



第十一章　SAP HANA OpenData 服务

第一节　SAP HANA 扩展应用服务

传统的数据库程序应用都是使用接口，如 ODBC、JDBC，来调用 SQL 语句来进行数据的管理(添加，更新，删除)或者数据查询，而对于数据库来讲，其大部分时间仅仅提供了数据的存储功能。如图 11-1 所示，传统的数据库应用分为客户端、应用服务器以及数据库三层(见图 11-1)。在传统数据库应用中，由于数据库不会对原始数据(Raw Data)进行过滤或者聚合，因此导致大量的数据被直接传输到应用服务器进行运算。这种情况有两个非常大的弊端，其一是数据量大的时候，应用服务器不得不等待所有原始数据全部传输过来，其二是应用服务器在处理大量数据时需要更多地占用系统的性能，当基于应用服务器的众多应用程序同时调用大量数据时，会造成系统运行缓慢，性能低下，导致终端用户不得不等待更长的时间来得到运行结果。

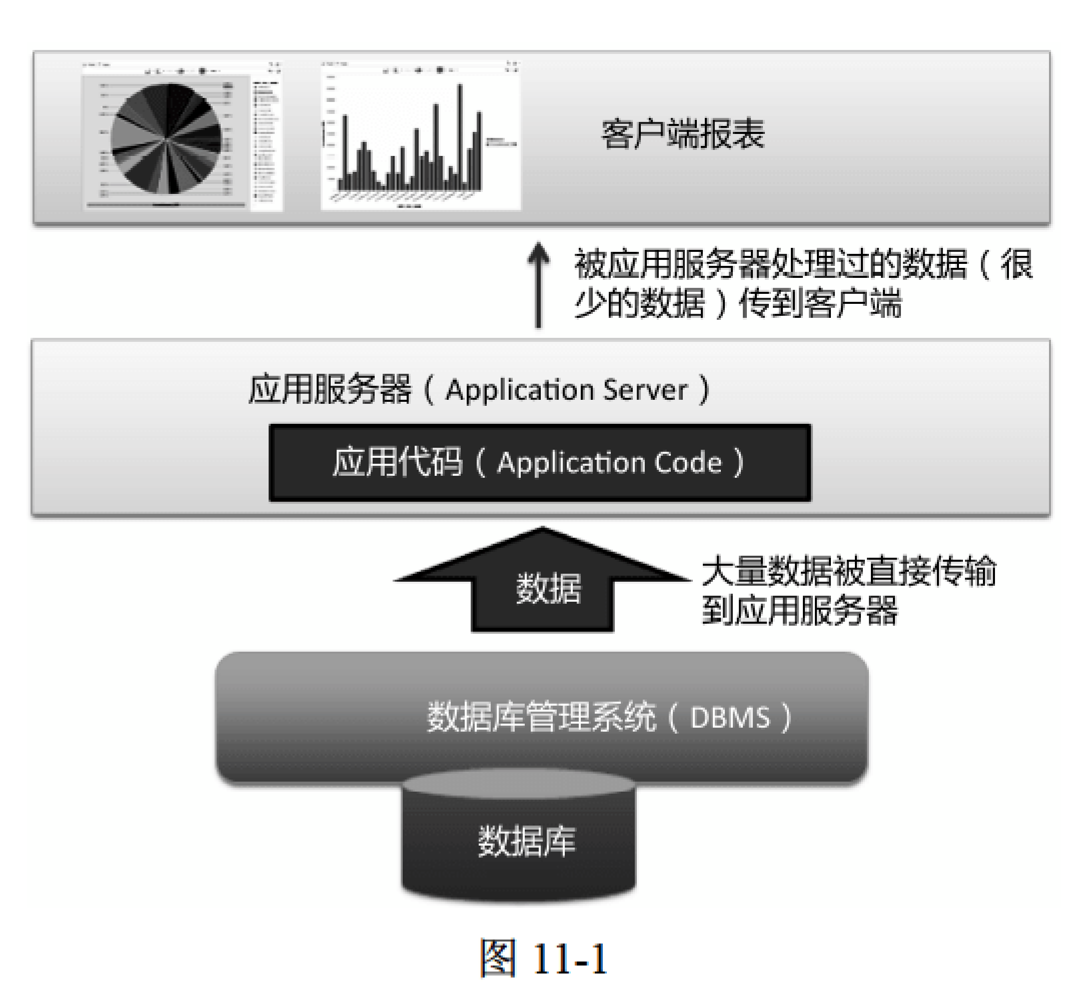


图 11-1



SAP HANA 通过大力扩展传统数据库服务器角色使上述情况得到很大的改善，其内置多种新功能的平台对于应用的开发和运行提供了广泛的支持。换句话说，你可以直接在 SAP HANA 数据库的基础上基于 SAP HANA 扩展平台开发和运行应用程序，而这种方式对于数据密集型程序尤其适用。我们将直接基于 SAP HANA 扩展平台开发的新应用统称为 SAP HANA 原生应用。SAP HANA 原生应用与传统数据库应用的最大不同在于它可以非常好地利用 SAP HANA 内存计算技术和数据库服务器并行计算能力来提高运行效率。

由图 11-2 我们可以看出，SAP HANA 原生应用在构架上通过将应用服务器下沉到 SAP HANA 平台，使得其与数据库连接得更加紧密。这样做的好处不仅是应用和数据库整合带来的巨大的性能提升，同时也精简了客户端接口和数据控制逻辑之间的中间层。

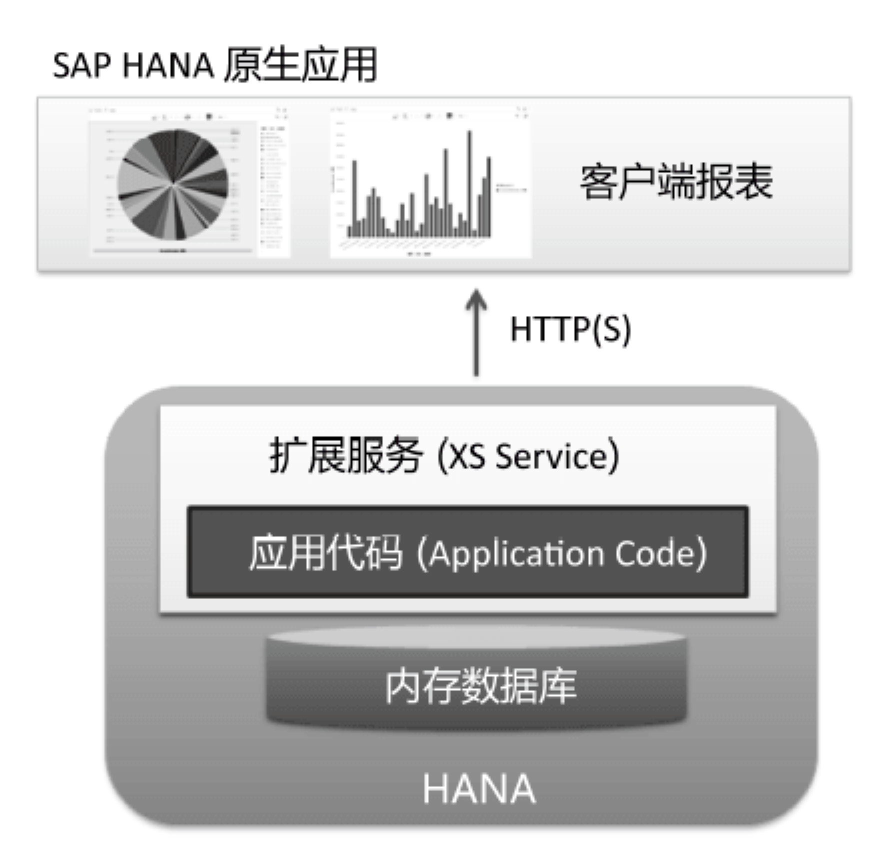


图 11-2

为了更好地支持这一构架，并为基于网络的应用提供端对端的支持，SAP HANA 提供了扩展应用服务(Extended Application Services)来提供很多嵌入式服务，这些服务包括轻量级网络服务器，可配置的 OData 服务支持，服务器端 JavaScript 运行服务，全部 SQL 以及 SQL Script 调用服务等。

上面讲述的所有 SAP HANA 扩展应用服务都是由 SAP HANA 扩展服务器(SAP HANA XS Server)提供的。SAP HANA 扩展服务器同时还能作为轻量级的，与 SAP HANA 完全整合的应用服务器，使得客户端应用可以通过 HTTP 协议直接访问 SAP HANA 的数据模型，视图以及数据表和存储过程而不需要像传统的数据库应用那样使用额外的应用服务器来控制应用与数据库的数据交换。值得一提的是，你可以通

过基于 SAP HANA 扩展服务器的 OData 服务快速建立动态的 HTML5 应用程序，稍后我们将详细介绍如何来实现它。

SAP HANA 扩展服务器更加完善了 SAP HANA，使其能够作为一种综合性的平台。通过 SAP HANA 扩展服务器，开发者可以基于 SAP HANA 平台对数据库的表、视图、存储过程等各方面编写服务器端的嵌入式程序。并且由于数据库服务器、应用服务器、网络服务器都在一台物理服务器上，因此能大大节省服务器维护及更新的工作量。

第二节　SAP HANA XS-OData 服务

OData，全称为开放数据协议(Open Data Protocol)，是来查询和更新数据的一种网络协议。如图 11-3 所示，OData 运用且构建于很多 Web 技术之上，比如 HTTP、Atom Publishing Protocol(AtomPub)和 JSON，其主要目标就是定义一种抽象的数据模型和协议，并将它们组合起来使得任何客户端可以访问任何数据源暴露出来的数据。我们这里所说的任何客户端是指诸如网页浏览器、移动设备、商业智能分析工具以及客户自主开发的程序(例如使用 PHP、Java 等语言开发的程序)，任何数据源是指诸如数据库、文件管理系统、云存储等数据源。OData 数据模型是指基于实体数据模型来管理和表述数据，OData 协议是指通过 OData 定义的查询语句来对数据进行生成、读取更新以及删除操作。OData 库是指在客户端预置的库文件以用于申请和展示 OData 数据。

更多详细信息，请参考 OData 官方网站：http://www.odata.org/。

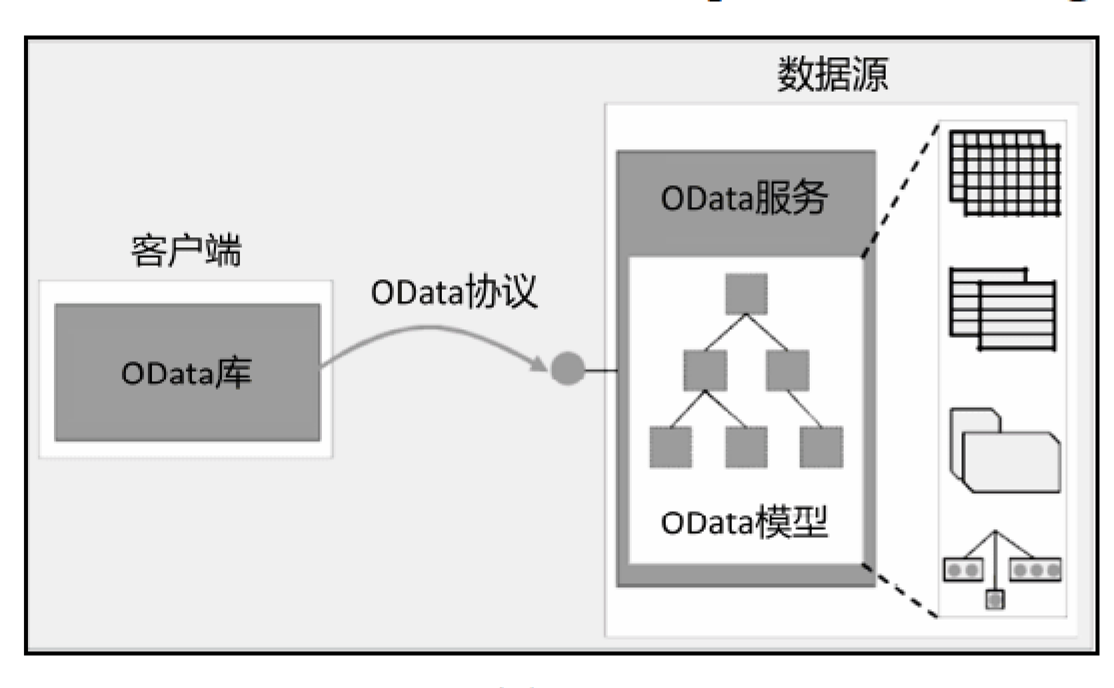


图 11-3



前面我们说过，SAP HANA 扩展应用服务器是提供 OData 服务支持的，那我们怎么通过 OData 服务来访问 SAP HANA 呢？下面我们通过一个案例来一步一步定义一个 OData 服务。在实际操作前，我们先来了解一下 SAP HANA 中 OData 服务的定义机制。所谓 OData 服务的定义，就是指你需要定义 OData 服务是用来暴露哪些数据，如何暴露以及给谁暴露的。在 SAP HANA 扩展应用服务器中，我们通过以后缀名为.xsodata 的文本文件来定义 OData 服务，这个文本文件至少要包含“service {}”条目，即便大括号中内容为空，这个文本文件在 SAP HANA 扩展应用服务器中也能生成一个完整的可执行 OData 服务，但是这个服务的目录和元数据文件都是空的。但是通常来讲，你需要在大括号中定义需要暴露的数据源，如数据表、SQL 视图或者计算规则等。

一、为定义 OData 服务准备基础环境

在定义 OData 服务前，我们需要保证下列条件都被满足：

- SAP HANA Studio 和 SAP HANA Client 已经安装并配置。
- 相关的用户权限已经赋予，例如创建新的包。
- SAP HANA Studio 已经连接上任一 SAP HANA Server。
- 你已经创建好了工作区、包结构和共享的项目。
- 你已经创建好了需要暴露的数据源，数据源至少包含 Schema 和数据表。

我们在前面的章节中已经讲述了如何安装/配置 SAP HANA Studio 和 SAP HANA Client 以及如何分配用户权限，那么接下来我们会从如何创建工作区(Workspace)和项目(Project)开始，到创建 Schema 和数据表，然后进行我们的 OData 服务创建。



1. 创建工作区

(1) 打开 SAP HANA Studio，进入“SAP HANA Development”透视图(见图 11-4)。

(2) 面板切换区域共有三个常用视图，它们是“Project Explore”、“SAP HANA Repositories”以及“SAP HANA System”。“Project Explore”视图用来生成和显示存储于本地的项目对象，“SAP HANA Repositories”视图则用来显示在服务器端的仓库中所包含的项目对象。“SAP HANA System”视图我们在前面多次介绍过了，这里不再重述。

首先我们切换到“SAP HANA Repositories”视图来生成工作区域(见图 11-5)。

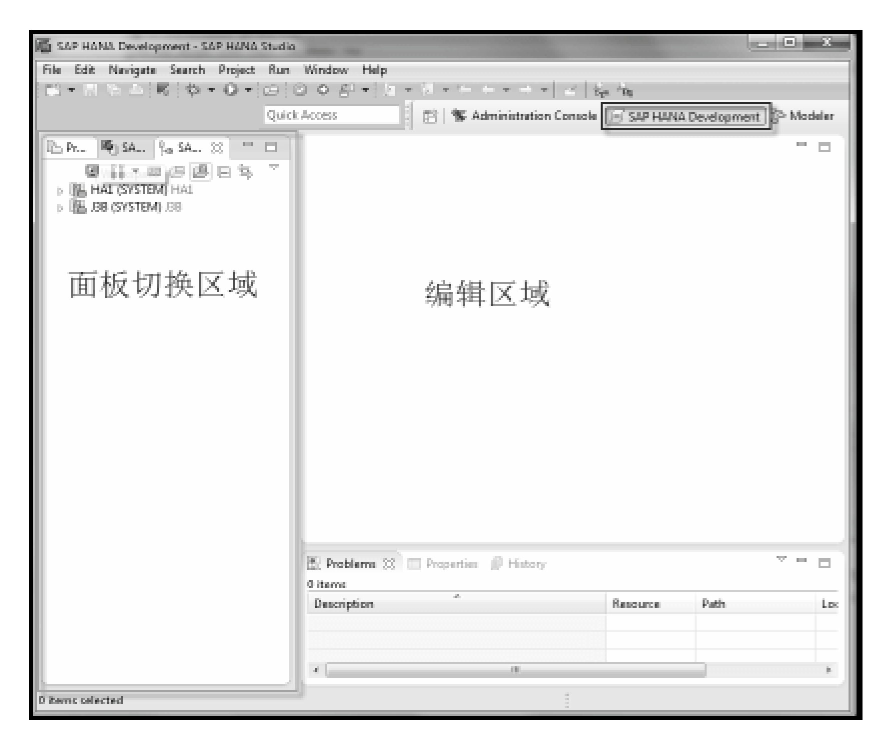


图 11-4

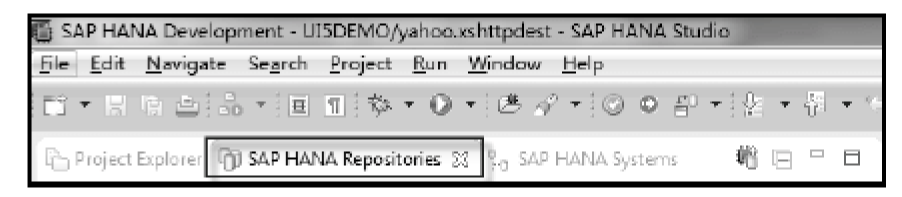


图 11-5

(3) 单击“Create New Repository Workspace”按钮，如图 11-6 所示。

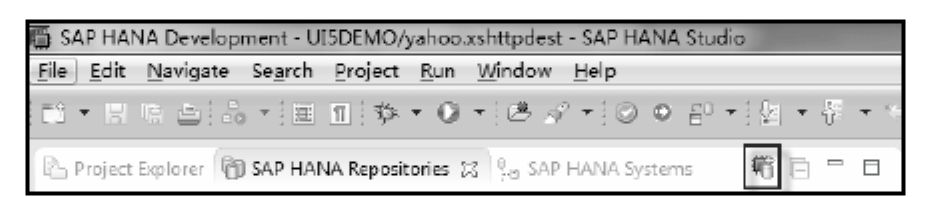


图 11-6

(4) 在“SAP HANA Systems”面板中选择 SAP HANA 系统(多系统情况下)。在“Workspace Name”字段中为你的新工作区输入名称，在“Workspace Root”字段中指定工作区的存储位置，单击“Finish”按钮(见图 11-7)。

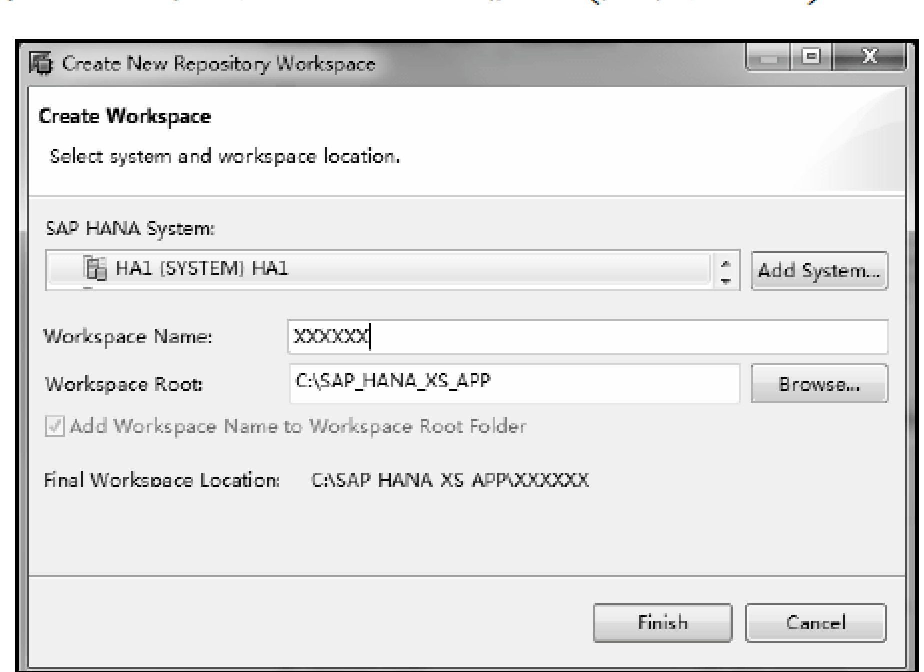


图 11-7



(5) 在“SAP HANA Repositories”视图中，你能找到刚刚创建的工作区(见图 11-8)。

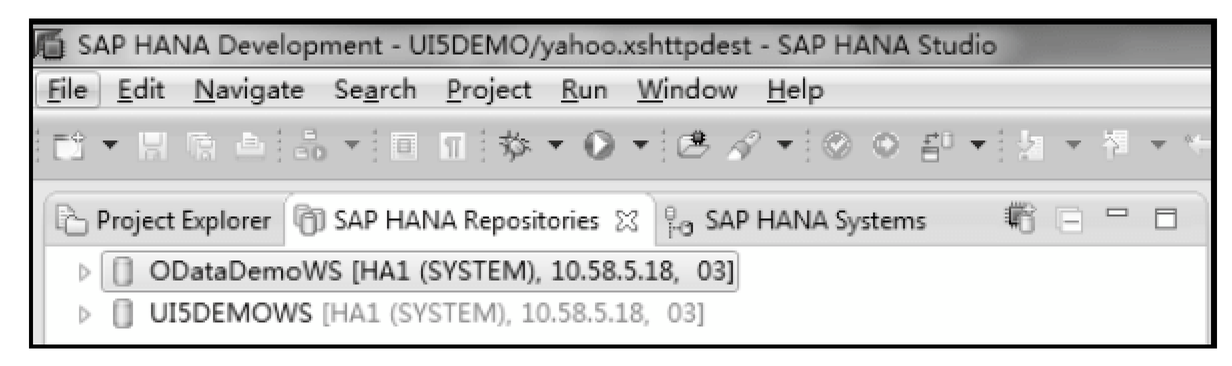


图 11-8

与此同时，在你指定的路径下(本例我们指定了“C:\SAP_HANAOData_Demo_WS”)，SAP HANA Studio 会同样生成一个文件夹用来存储本地的项目对象(见图 11-9)。

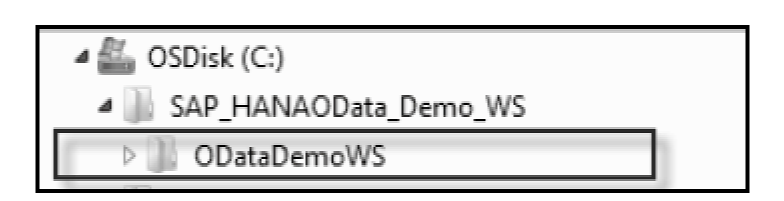


图 11-9

到目前为止，你已经成功创建了工作区。在 SAP HANA Studio 中，工作区一是用来指定你的项目文件是存储在本地哪个路径下面，二是用来与 SAP HANA 仓库进行文件同步。创建完工作区后，接下来我们需要创建一个项目来包含所有需要创建的 OData 服务对象。

2. 创建项目

(1) 这次我们切换到“Project Explorer”面板，单击“New”菜单，在下拉列表中选择“Project”一项(见图 11-10)。

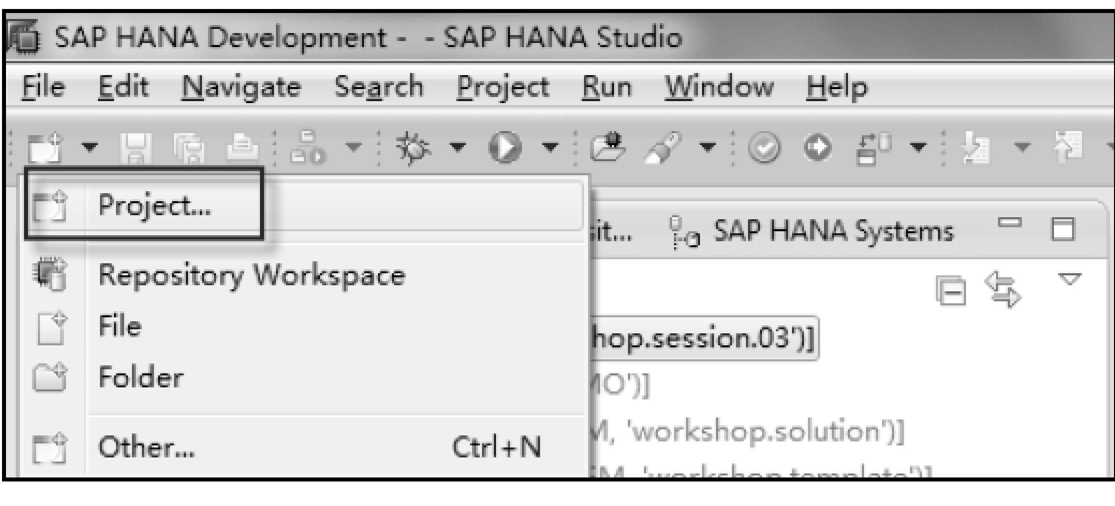


图 11-10



(2) 在弹出的“New Project”窗口中展开“SAP HANA Development”节点，选择“XS Project”并单击“Next”按钮(见图 11-11)。

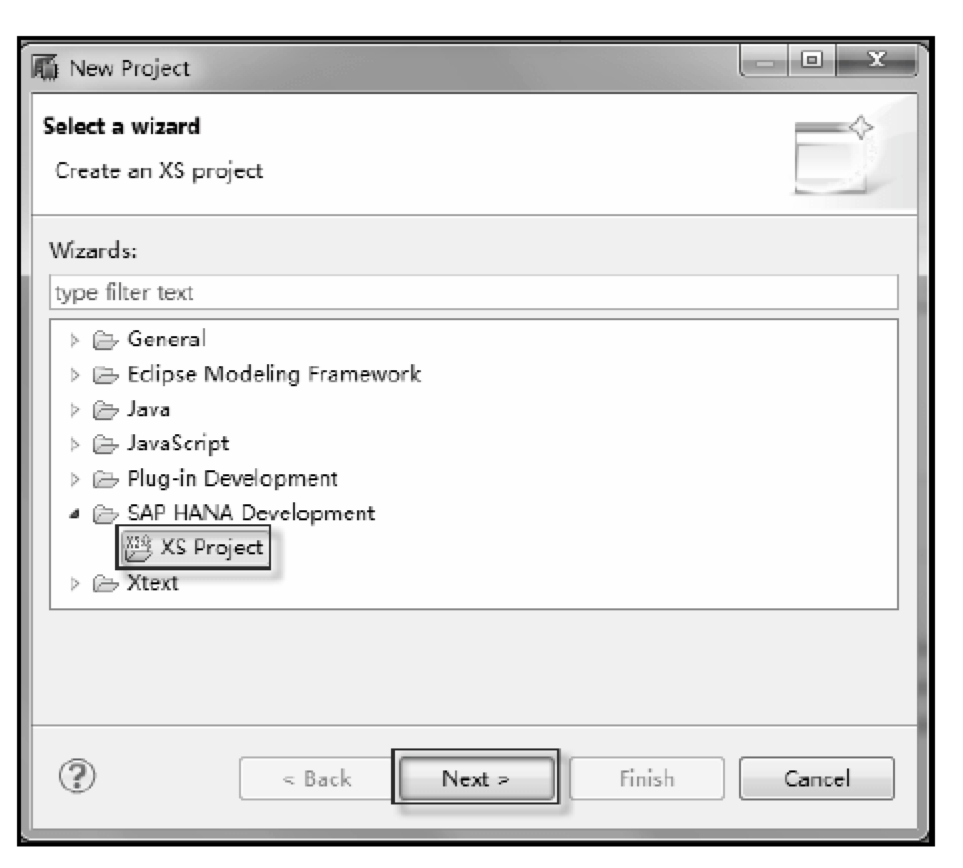


图 11-11

(3) 在“Project name”字段输入项目名称。在同一个工作区，项目名称是唯一的。最后我们单击“Finish”按钮确认(见图 11-12)。

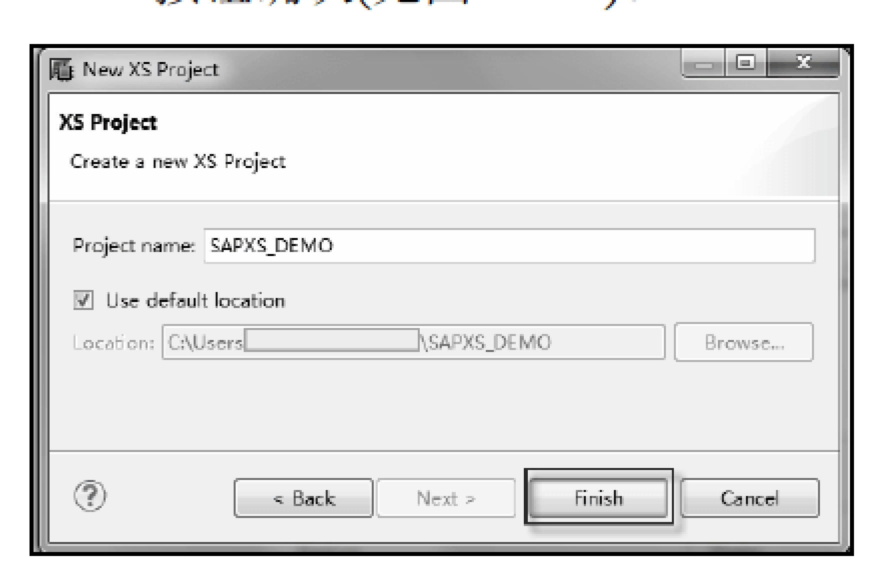


图 11-12

(4) 在“Project Explore”视图中检查新建的项目是否已经存在(见图 11-13)。

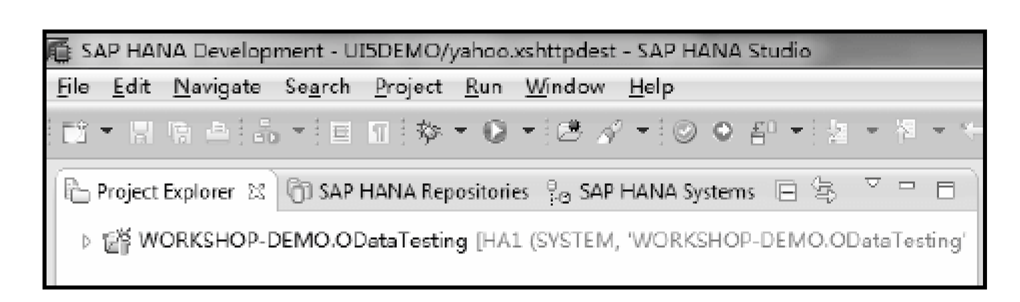


图 11-13

3. 共享项目

在创建完项目后，你必须将这个项目与之前创建的工作区关联起来，这个操作我们称之为共享项目。通过共享项目，你保存在本地的项目文件能够被同步并存储

到远程的 SAP HANA Server 仓储中。共享项目除了能把项目和 SAP HANA 仓储关联起来外，同时能保证你的其他团队成员都能够访问共享的项目。

(1) 在“Project Explore”视图中，右击刚刚生成的项目“WORKSHOP-DEMO.ODataTesting”，选择“Team”→“Share Project”(见图 11-14)。

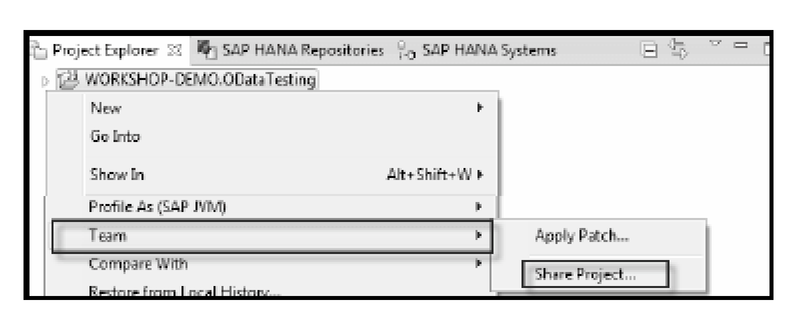


图 11-14

(2) 在“Share Project”窗口中，选择“SAP HANA Repository”，单击“Next”按钮(见图 11-15)。

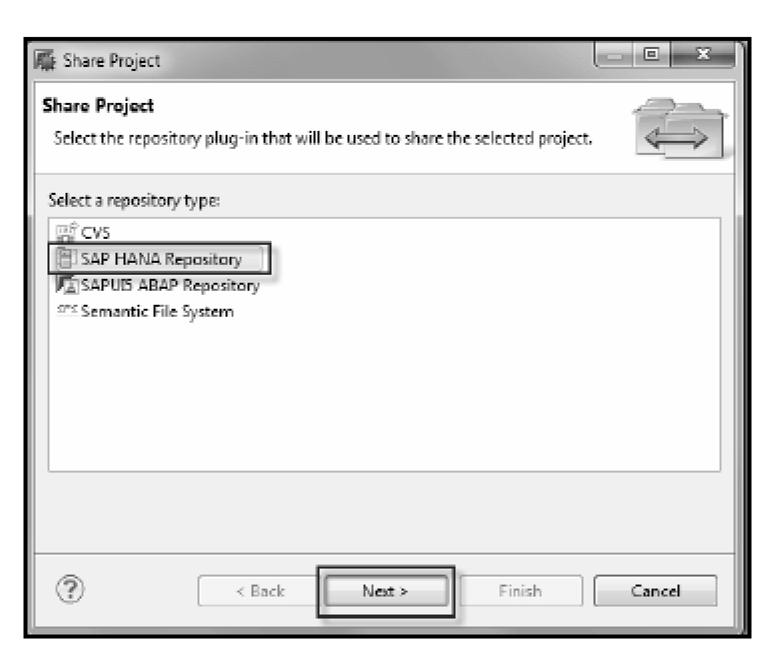


图 11-15



(3) 指定需关联的工作区并检查其他信息是否正确，最后点单击“Finish”按钮(见图 11-16)。

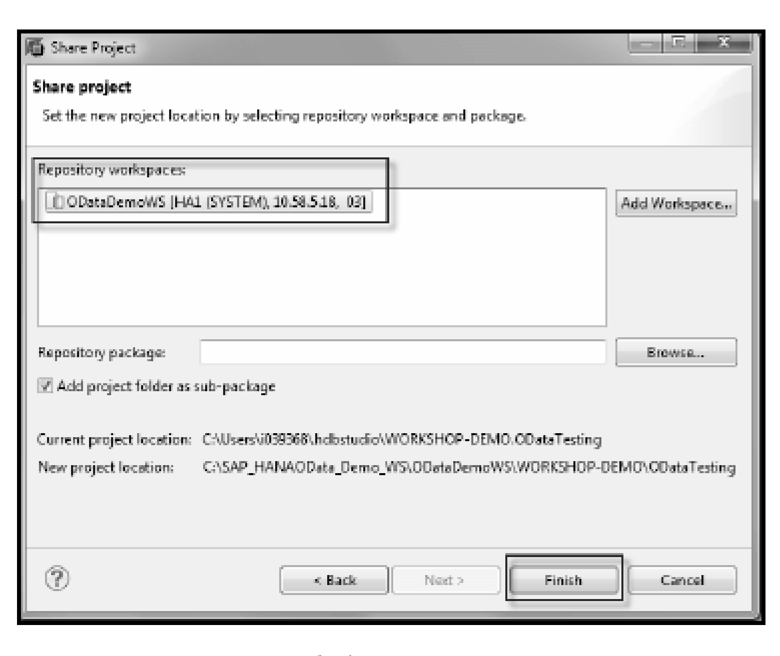


图 11-16

(4) 在“Project Explore”视图中检查共享后的项目状态(见图 11-17)。

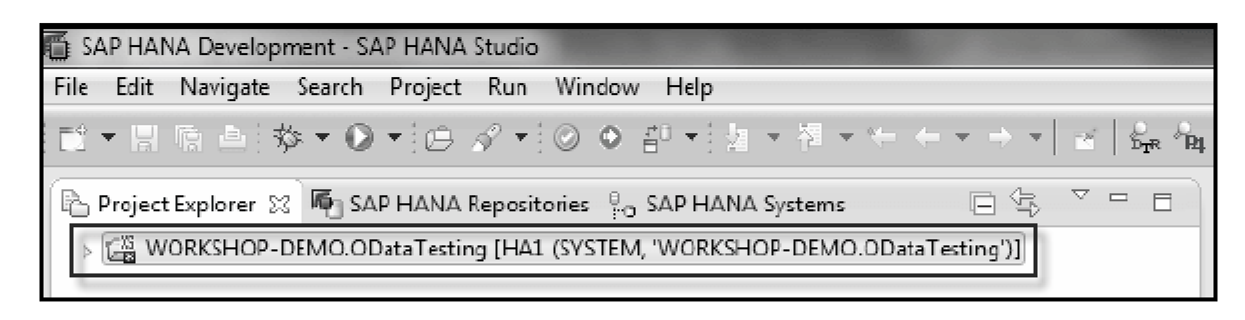


图 11-17

从图 11-17 我们可以看出，我们新建的项目已经和工作区关联在一起了。仔细看项目的图标，你会发现图标的右下角有一个“*”的标志，这个标志表示项目中的文件已经发生改变，但是这个改变还没有提交到远程服务器中 SAP HANA 知识仓库中去。如果我们把视图切换到“SAP HANA Repositories”视图，我们会发现在服务器端的“WORKSHOP-DEMO”包下是空的(见图 11-18)。

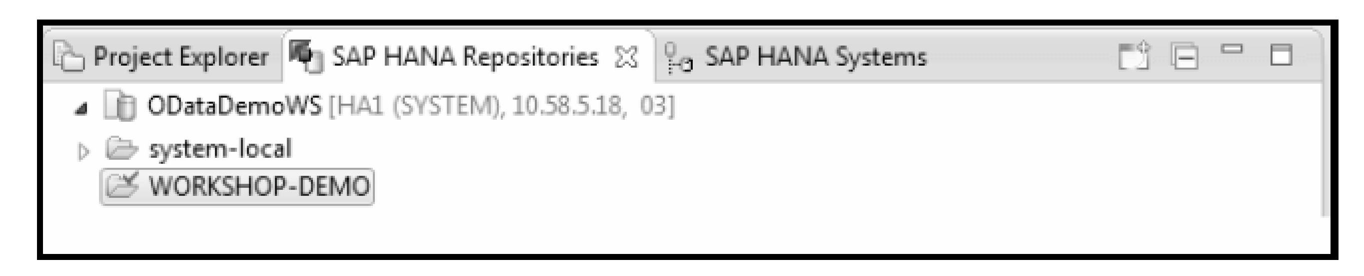


图 11-18

(5) 在“Project Explore”视图中，右击项目“WORKSHOP-DEMO.ODataTesting”节点，选择“Team”→“Commit”来提交修改后的项目。提交操作是用来将修改过的项目或者项目包含的对象添加到工作区的操作。项目或者项目包含的对象被提交后，你就能在“SAP HANA Repositories”视图中看见它们，但是在你激活它们之前你还不能使用它们(见图 11-19)。

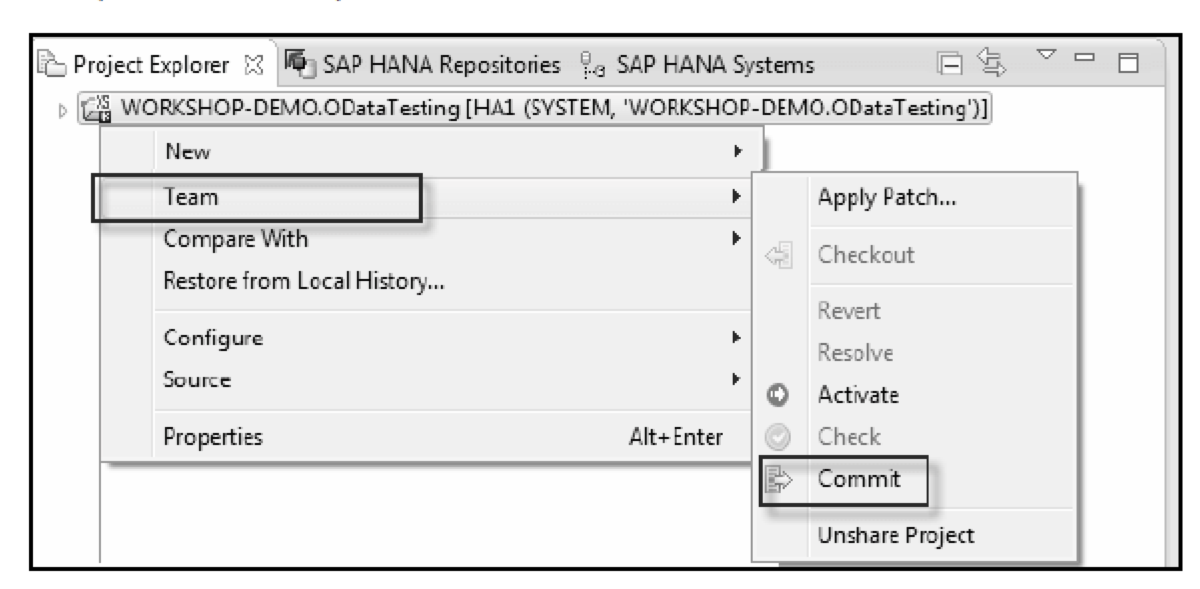


图 11-19

(6) 提交后我们切换回“SAP HANA Repositories”视图，右击“WORKSHOP-DEMO”节点并选择“Refresh”一项(见图 11-20)。





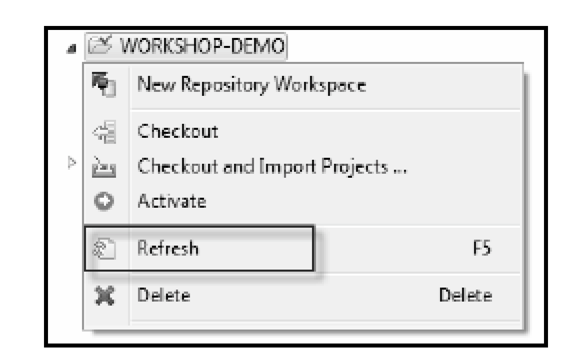


图 11-20

此时“WORKSHOP-DEMO”节点下面会出现我们刚建好的项目。你可以看到项目“project”的图标右下角为灰色菱形，代表了项目的最新版本已经存在于 SAP HANA 知识仓库中，但是还没有被激活(见图 11-21)。

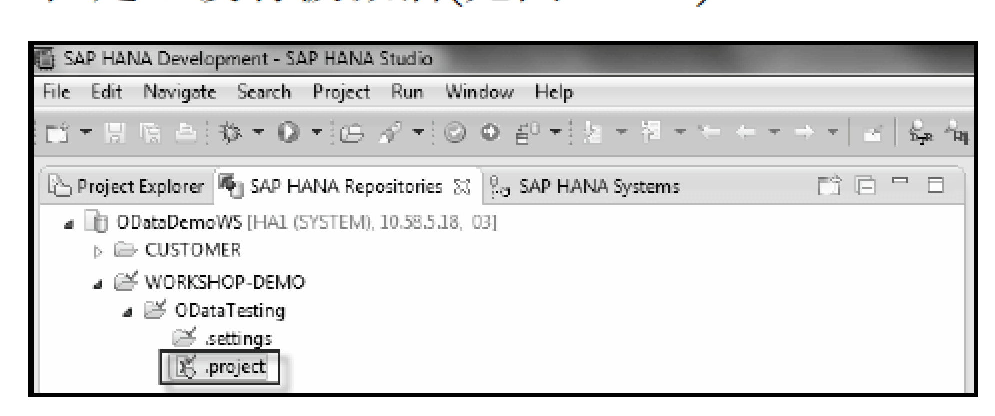


图 11-21

(7) 在“Project Explore”视图中，右击项目“WORKSHOP-DEMO.ODataTesting”节点，选择“Team”→“Activate”来在工作区中激活项目(见图 11-22)。

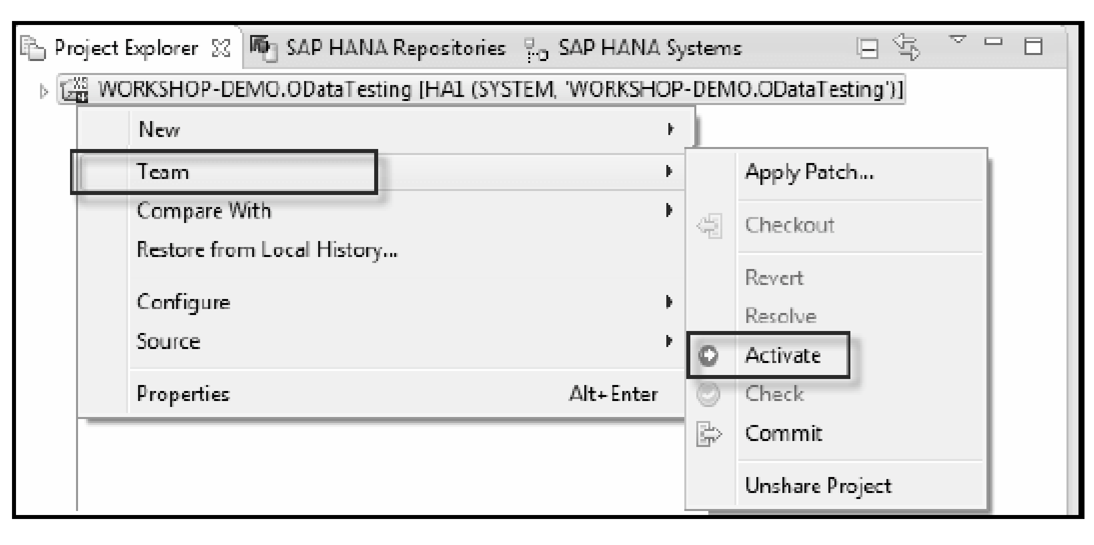


图 11-22

项目激活后你可以看见在“Project Explore”视图中项目图标右下角的“*”标志消失了(见图 11-23)。

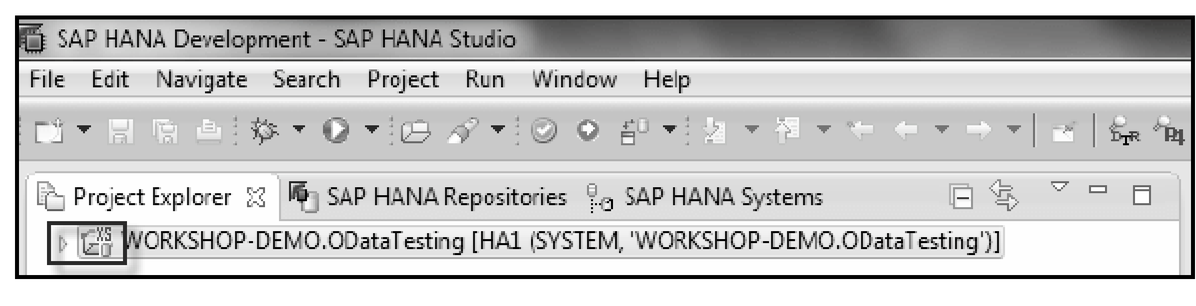


图 11-23



然后我们切换回“SAP HANA Repositories”视图，右击“WORKSHOP-DEMO”节点并选择“Refresh”(见图 11-24)。

你可以看见在“SAP HANA Repositories”视图中项目图标右下角的菱形也标志消失了，证明项目激活成功(见图 11-25)。

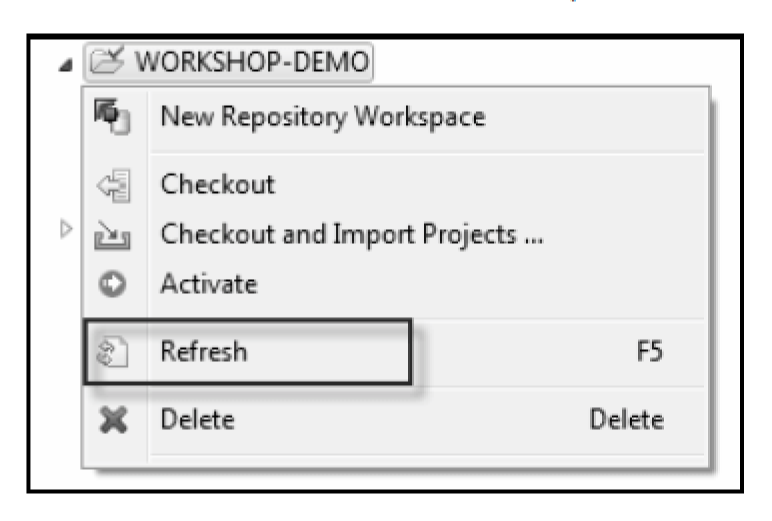


图 11-24

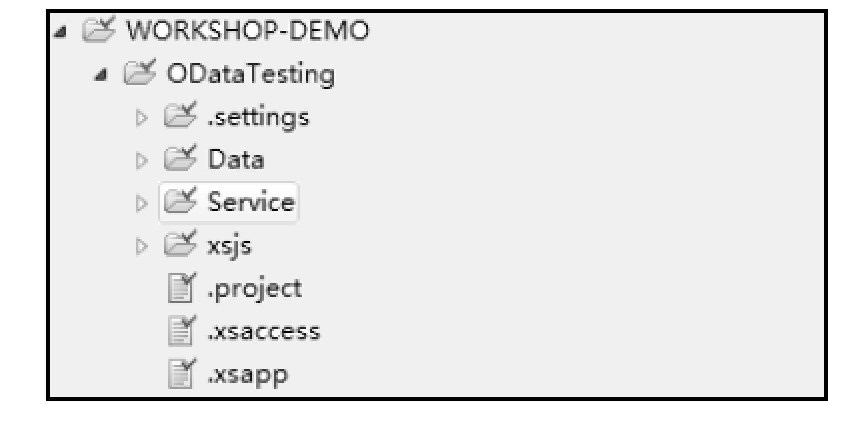


图 11-25

(8) 在“SAP HANA System”视图中检查对应包中的内容。如果你还有印象，我们在介绍如何配置 SAP HANA Studio 时建立的“WORKSHOP-DEMO”包是一个空的包。我们在新建项目的名称中引用了包名“WORKSHOP-DEMO”，并将项目和工作区完成关联且激活后，系统会自动在“WORKSHOP-DEMO”包下面为我们的项目新建一个以项目名为名称的子包“ODataTesing”(见图 11-26)。

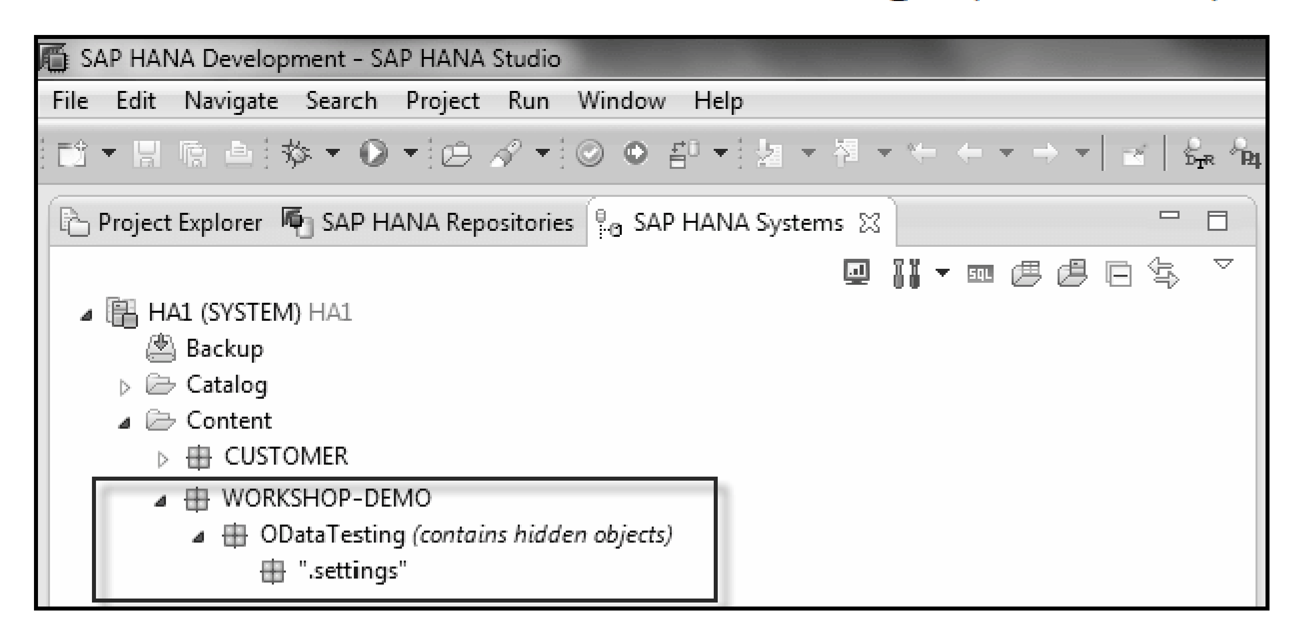


图 11-26

至此，我们完成了基础环境的配置。我们新建了工作区，新建了项目，并把工作区和项目关联了起来，然后我们完成了提交和激活的操作。这些操作流程非常有用，因为在今后你开发基于 SAP HANA 的原生程序时，你也要遵循一样的流程来配置好你的基础开发环境。那么接下来我们要做的就是为 OData 服务准备需要暴露的数据源(Data Source)。



二、为定义 OData 服务准备数据源

1. 创建文件夹

为了更好地归类项目中创建的各种文档，我们通过创建文件夹的方式来分别储存这些文档，例如我们可以创建“Data”文件夹来放置数据类型的文档，而“Service”文件夹则可以放置服务类型的文档等。

(1) 在“Project Explore”视图中右击项目“ODataTesting”，在弹出的右键菜单中选择“New”→“Folder”(见图 11-27)。

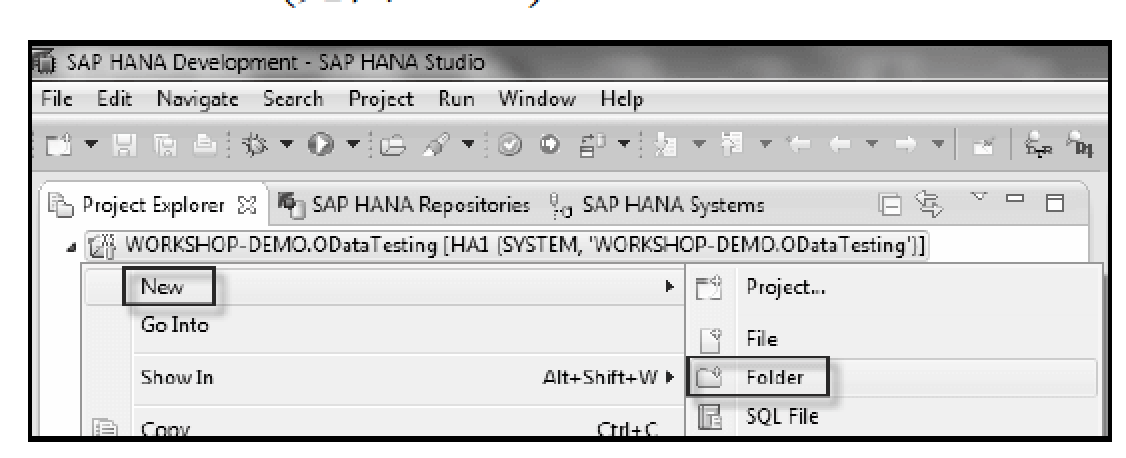


图 11-27

(2) 在弹出的“New Folder”窗口中的“Folder Name”字段输入文件夹名称，本例中我们输入“Data”来表明这个文件夹是用来存储数据类型的文档的。单击“Finish”按钮确认(见图 11-28)。

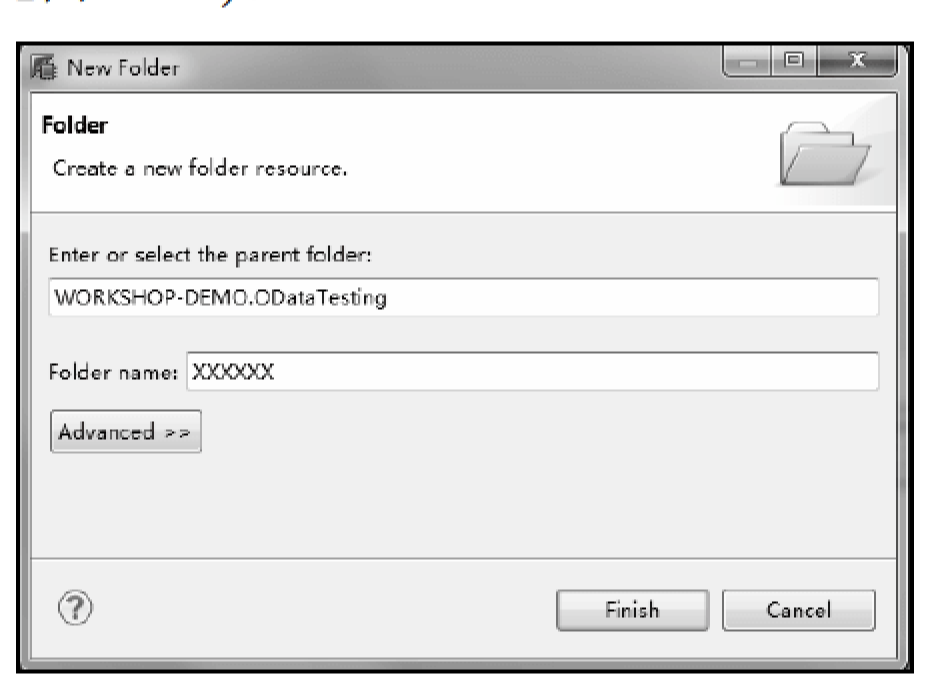


图 11-28



(3) 提交和激活新的文件夹。右击“Data”文件夹，选择“Team”→“Commit”/“Activate”(见图 11-29)。

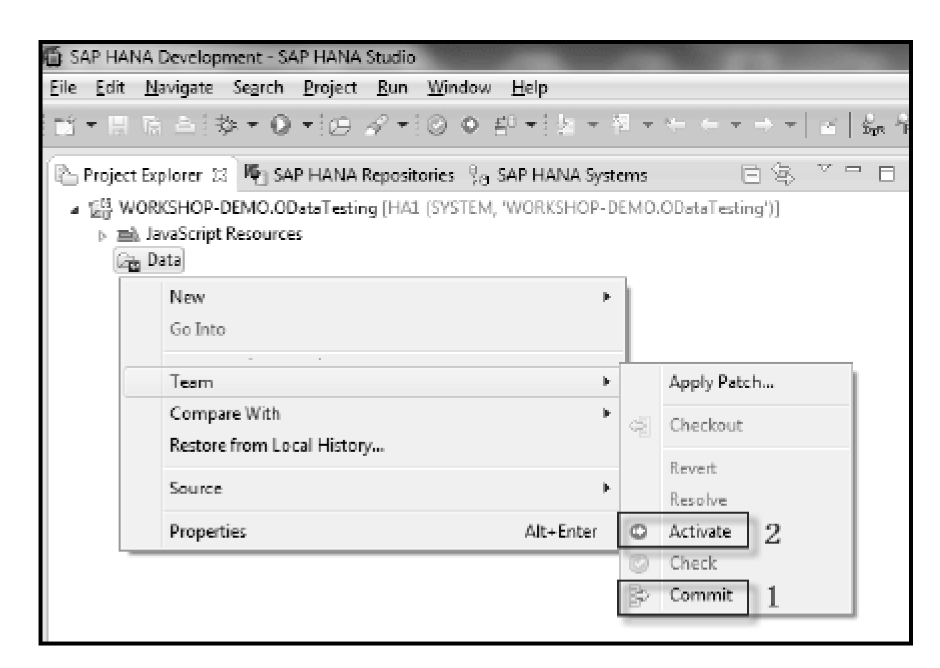


图 11-29

2. 创建节点

创建好“Data”文件夹后，我们就需要为 OData 服务数据源准备数据存储相关的文件了，首先我们需要创建一个 Schema 文件来存放用来存储数据用的数据表的结构。

(1) 创建 Schema 定义文件。右击“Data”文件夹，选择“New”→“File”(见图 11-30)。

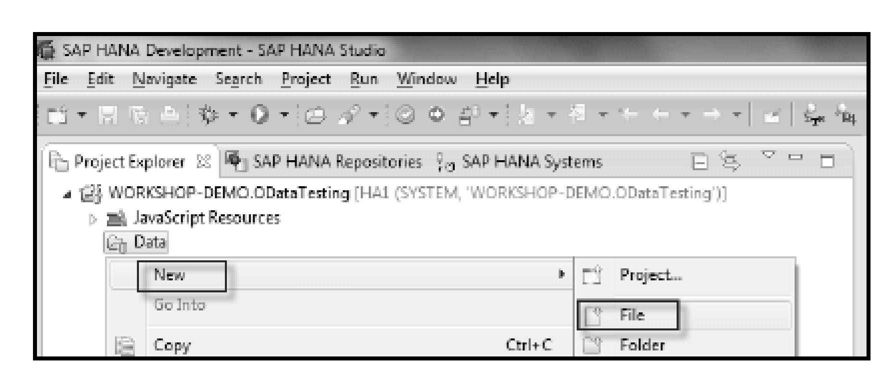


图 11-30

(2) 在弹出的“New File”窗口中的“File Name”字段输入 Schema 定义文件的名称。在 SAP HANA Stuio 中，所有创建 Schema 的文件都是以“.hdbschema”作为后缀名的。输入完名称后，单击“Finish”按钮确认(见图 11-31)。

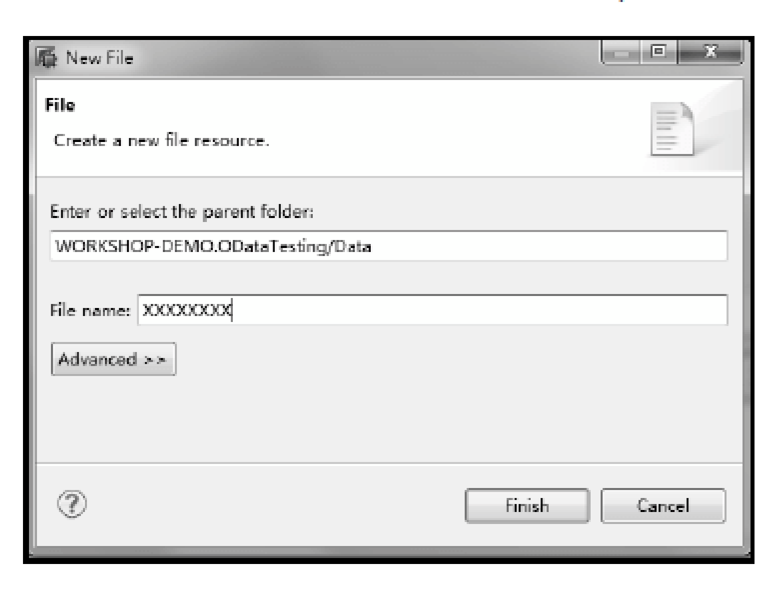


图 11-31

(3) 在创建好 Schema 定义文件后，你能在 SAP HANA Stuio 界面右侧的编辑区来定义 Schema 的名称。在编辑区输入 Schema 的定义代码“schema_name="ODATATEST";”并按保存按钮保存输入。在 SAP HANA Studio 中，定义 Schema 的代码通用格式为“schema_name="MYSCHEMA";”。需要注意的是，请使用大写字母来命名新的 Schema，如果你使用小写字母的话将会导致未知的错误(见图 11-32)。

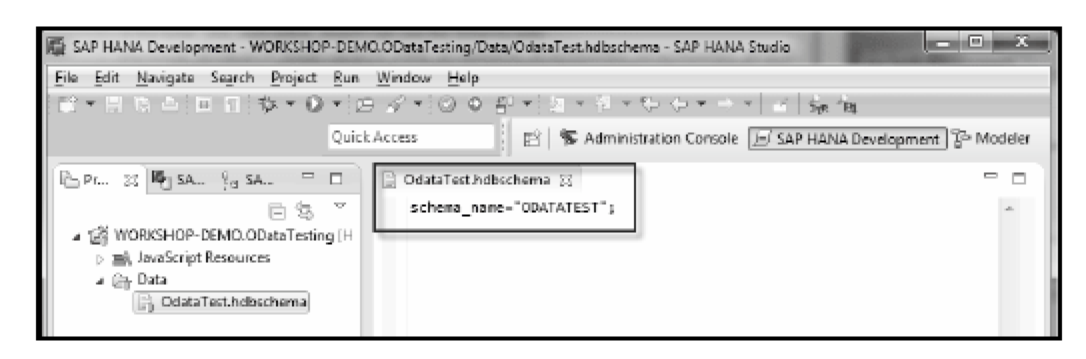


图 11-32

(4) 提交并激活 Schema 的定义文件。右击 Schema 定义文件，选择“Team”→“Commit”/“Activate”(见图 11-33)。

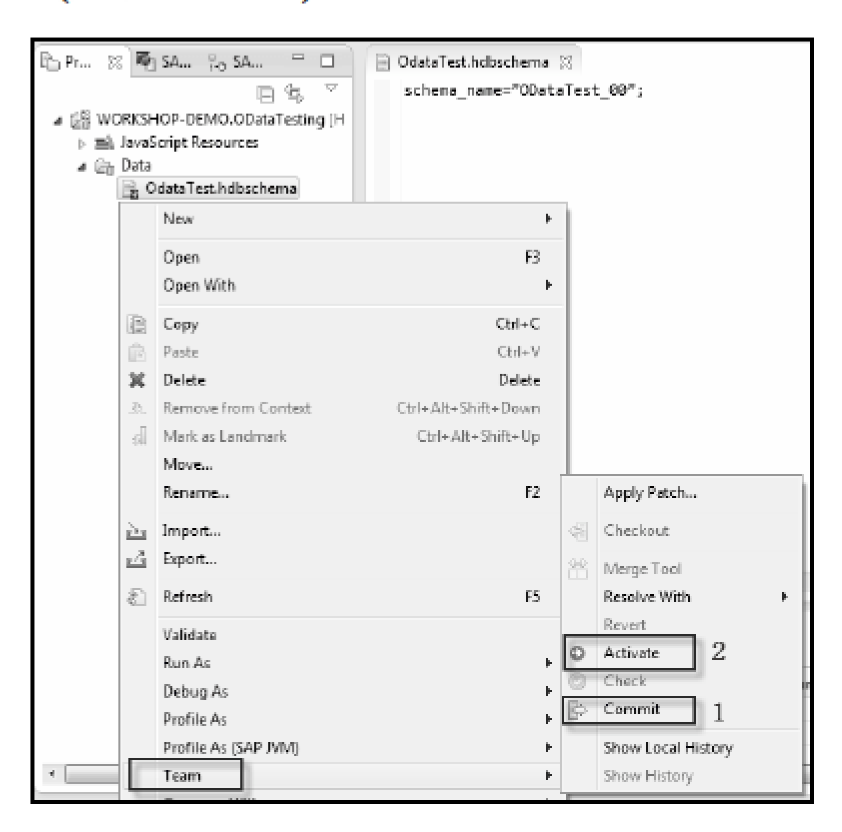


图 11-33

(5) 在“SAP HANA System”视图中检查新生成的 Schema(见图 11-34)。

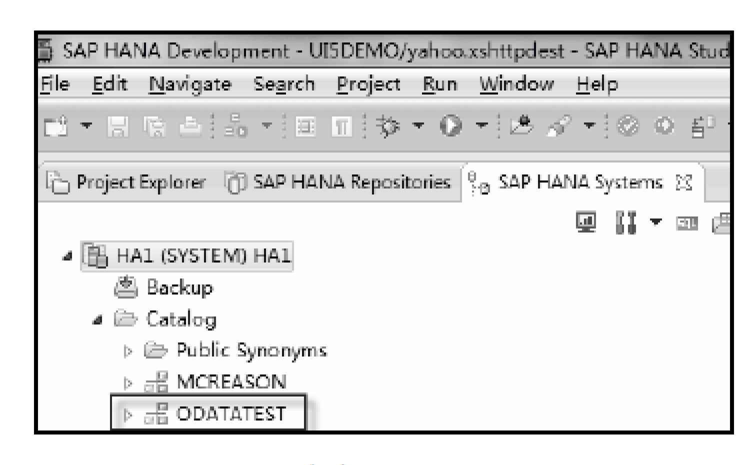


图 11-34



(6) 为用户赋予操作新 Schema 的权限。在激活 Schema 后，只有“_SYS_REPO”用户可以看到并访问新生成的 Schema，而其他用户由于权限的设置问题是看不到这个 Schema 的。接下来我们要做的就是为指定的用户授予权限使他们能够看见并访问新生成的 Schema。在“SAP HANA System”视图中定位到“Security”→“Users”→你要授予权限的用户，双击打开用户维护窗口(见图 11-35)。

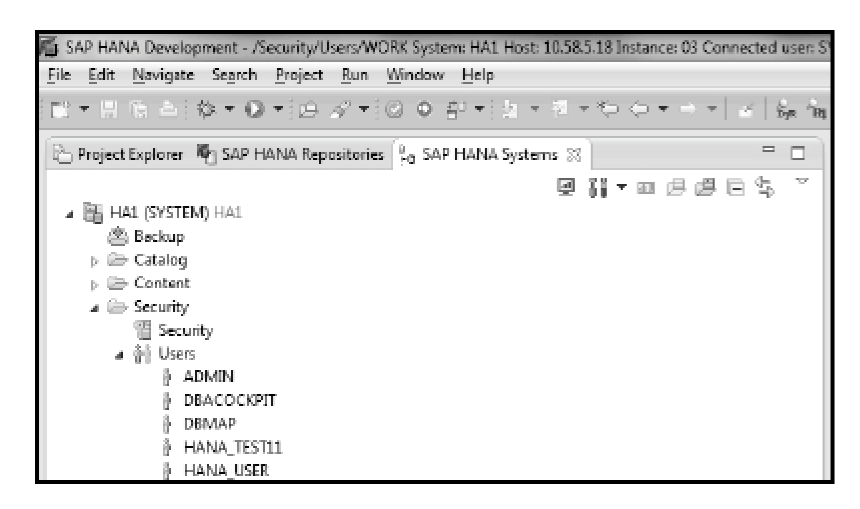


图 11-35

(7) 在用户维护窗口中打开“Object Privileges”标签页。单击“+”按钮添加权限(见图 11-36)。

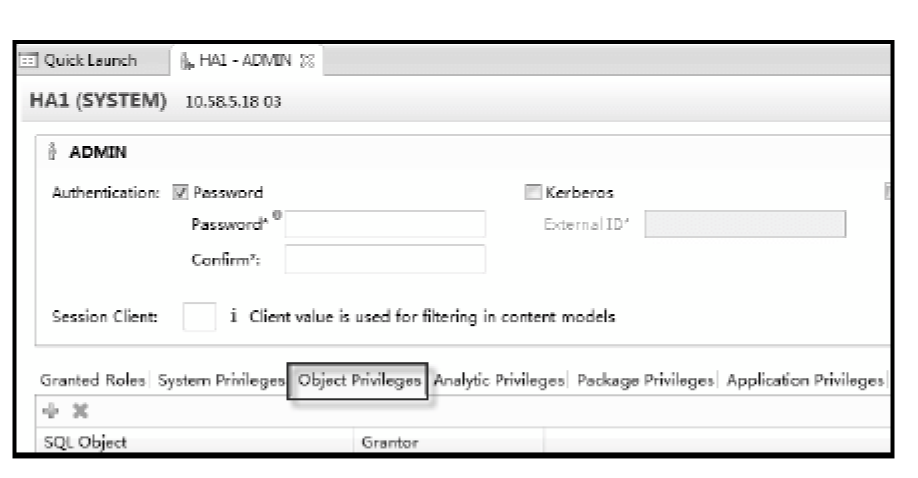


图 11-36

(8) 在弹出的“Select Catalog Object”窗口中按照关键字符搜索新的 Schema。选中符合的条目，单击“OK”按钮确认(见图 11-37)。

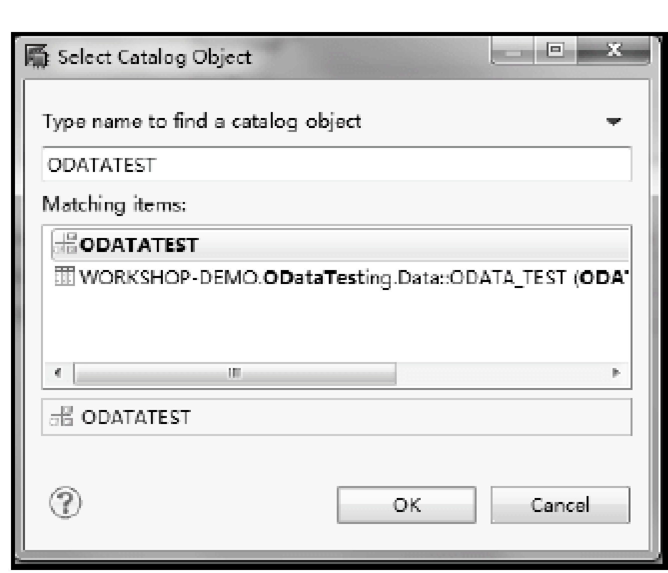


图 11-37

确认完成后你就能在“SQL Privileges”标签页下面找到需要授权的 Schema，“SQL Privileges”标签页表头上的叹号标志代表有未尽的授权操作(见图 11-38)。

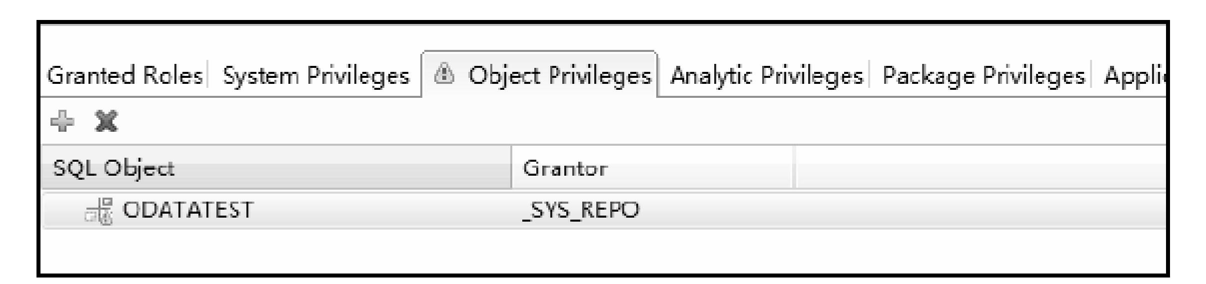


图 11-38

(9) 在右方的“Privileges for”面板下面通过选择单选框来授予用户对于“ODATATEST”Schema 的相应操作权限。“SQL Privileges”标签页表头上的叹号标志会随之消失(见图 11-39)。

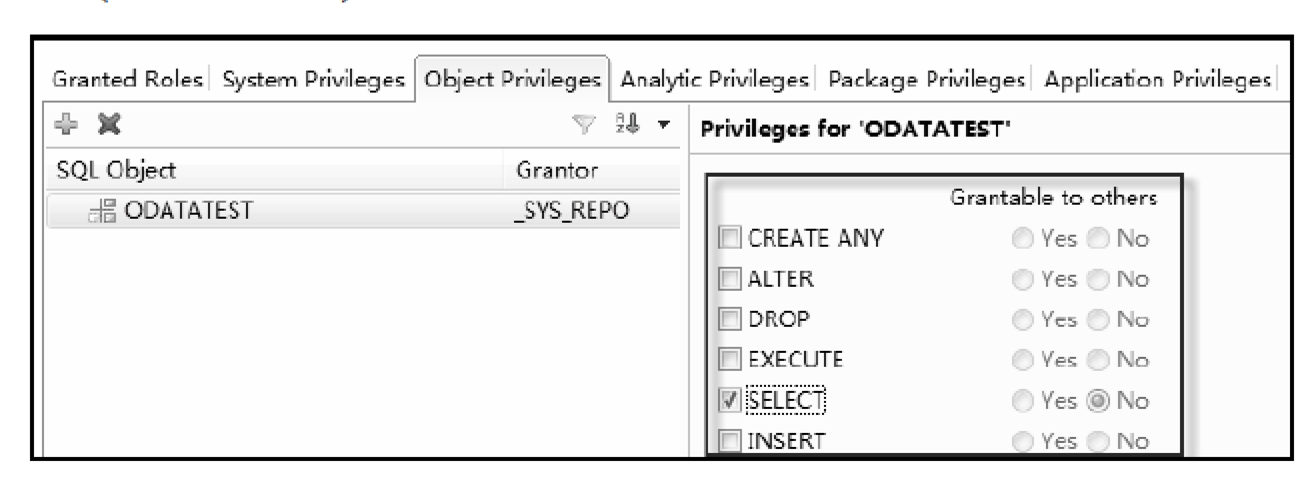


图 11-39

(10) 单击保存按钮，在用户维护界面的信息栏里面查看最终操作状态(见图 11-40)。

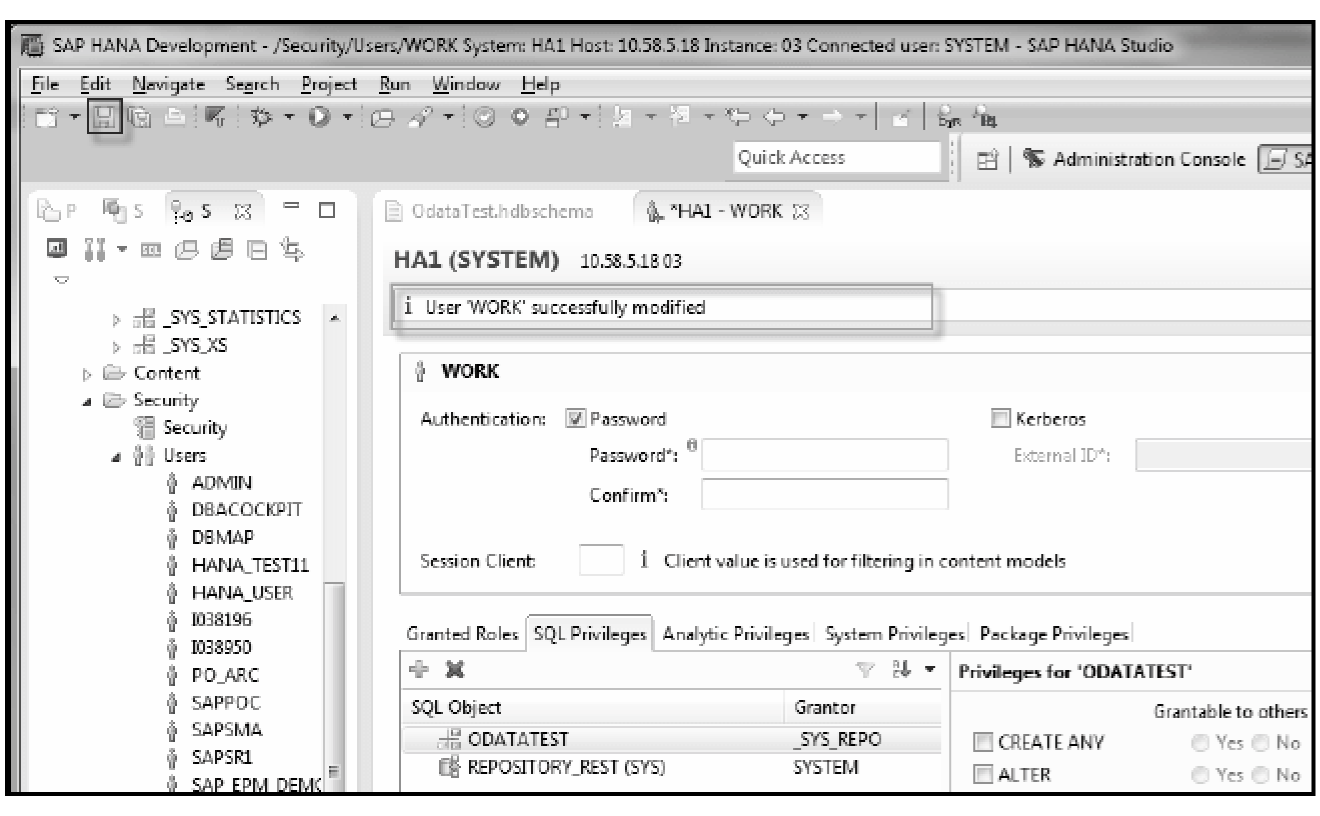


图 11-40

到此为止，我们已经成功地将新建的 Schema 的相应权限授权给了需要的用户，这样一来我们就可以在 Schema 中创建数据表和其他数据库对象了。

3. 创建数据表

接下来我们开始创建数据表。在 SAP HANA Studio 中，你如果要在项目中创建数据表，请参照下列语句：

```
    table.schemaName = "MYSCHEMA";
    table.tableType = COLUMNSTORE;
    table.columns = [
    {name = "Col1"; sqlType = VARCHAR; nullable = false; length = 20;
comment = "dummy comment";}, {name = "Col2"; sqlType = INTEGER;
nullable = false;},
    {name = "Col3"; sqlType = NVARCHAR; nullable = true; length = 20;
defaultValue = "Defaultvalue";}, {name = "Col4"; sqlType = DECIMAL;
nullable = false; precision = 2; scale = 3;}];
    table.indexes = [
    {name = "MYINDEX1"; unique = true; indexColumns = ["Col2"];},
    {name = "MYINDEX2"; unique = true; indexColumns = ["Col1",
"Col4"];}];
    table.primaryKey.pkcolumns = ["Col1", "Col2"];
```

(1) 右击“Data”文件夹，选择“New”→“File”(见图 11-41)。

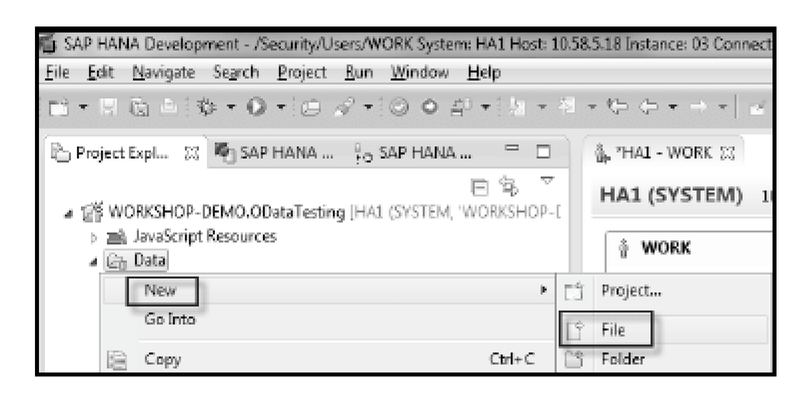


图 11-41

(2) 输入数据表定义文件名称，该名称必须以“.hdbtable”结尾，单击“Finish”按钮确认(见图 11-42)。

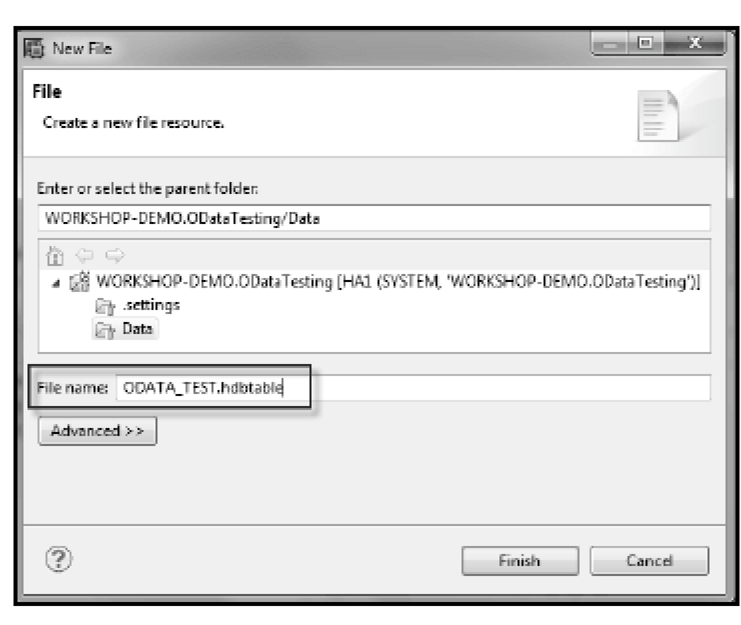


图 11-42

(3) 在数据表生成编辑界面，将如下代码输入到编辑区域：这个测试数据表包含了“客户代码”、“客户名称”以及“所在城市”三个字段，其中“客户代码”为表的关键字段。整体定义如下：

```
table.schemaName = "ODATATEST";
table.tableType = COLUMNSTORE;
table.description = "OData Service Test Table";
table.columns = [
{name = "CUSTOMER_ID"; sqlType = INTEGER; nullable = false;},
{name = "CUSTOMER_NAME"; sqlType = NVARCHAR; nullable = false; length = 60;},
{name = "CITY"; sqlType = NVARCHAR; nullable = true; length = 20; }
];
table.primaryKey.pkcolumns = ["CUSTOMER_ID"];
```

界面如图 11-43 所示。

```
ODATA_TEST.hdbtable
table.schemaName = "ODATATEST";
table.tableType = COLUMNSTORE;
table.description = "OData Service Test Table";
table.columns = [
{name = "CUSTOMER_ID"; sqlType = INTEGER; nullable = false;},
{name = "CUSTOMER_NAME"; sqlType = NVARCHAR; nullable = false; length = 60;},
{name = "CITY"; sqlType = NVARCHAR; nullable = true; length = 20; }
];
table.primaryKey.pkcolumns = ["CUSTOMER_ID"];
```

图 11-43

(4) 在项目中右击数据表定义文件，选择“Team”→“Commit”/“Activate”提交并激活数据表定义文件(见图 11-44)。

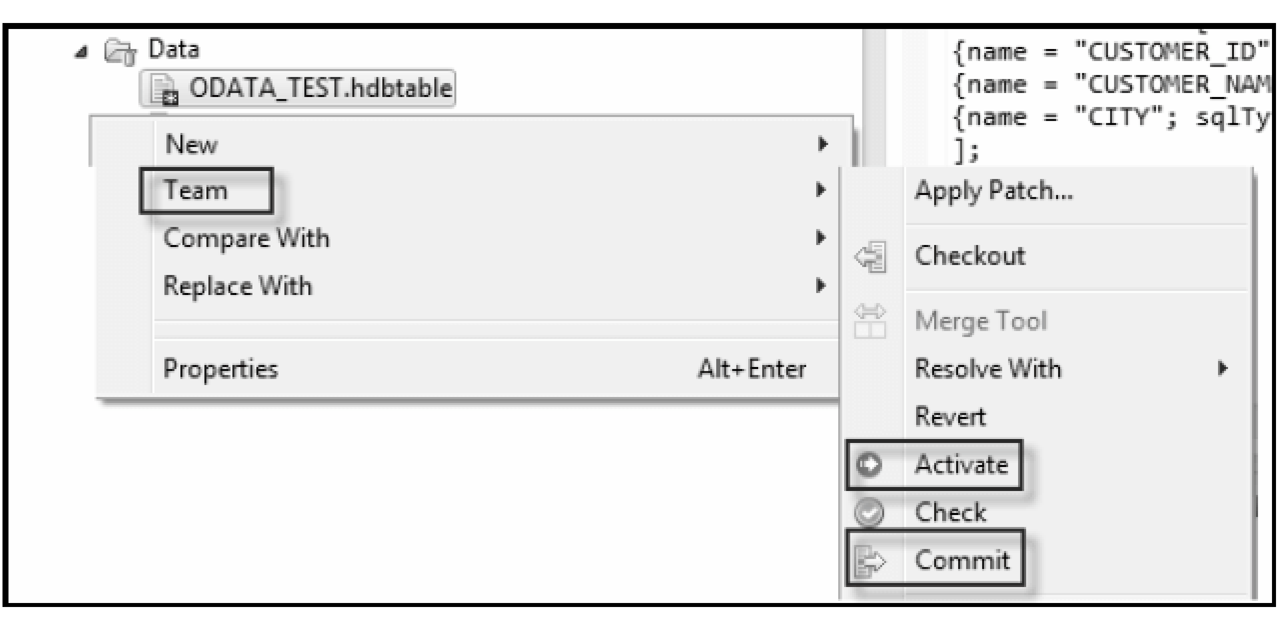


图 11-44

(5) 在“SAP HANA System”视图中检查新生成的数据表。



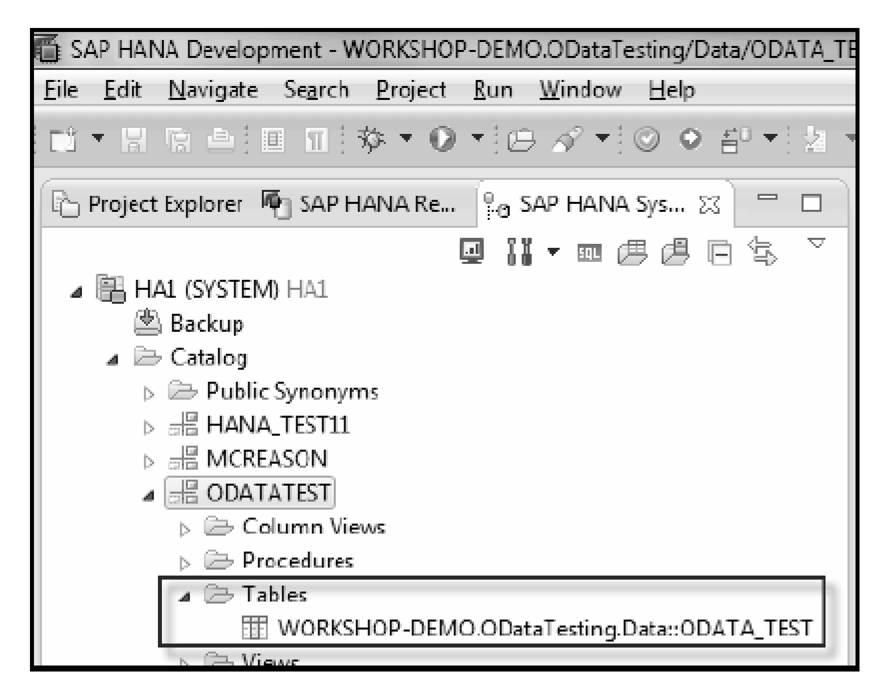
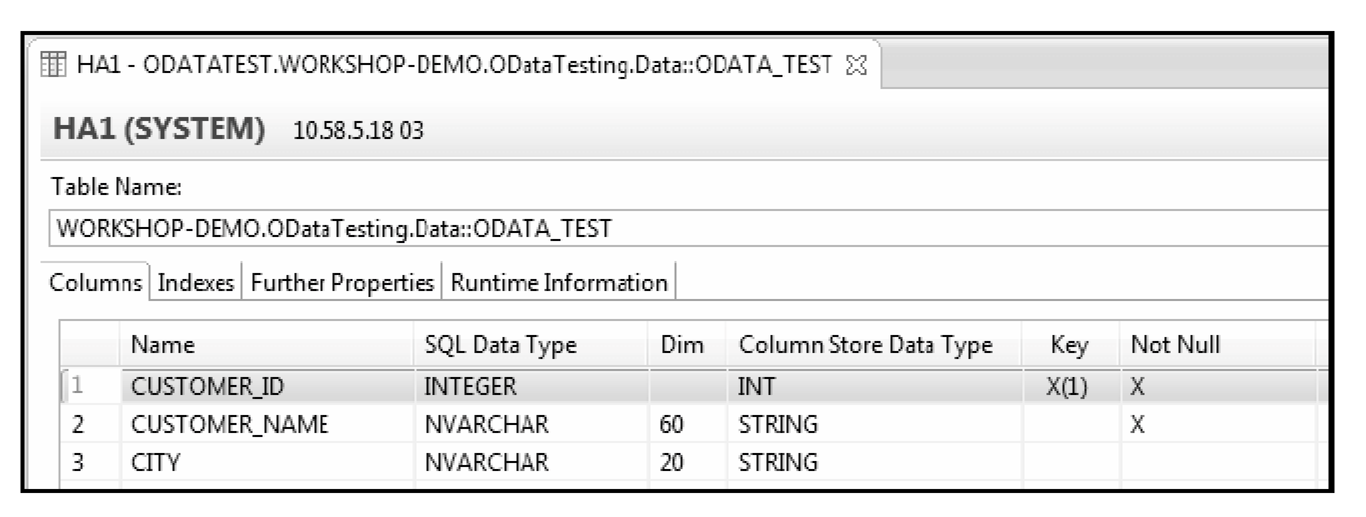


图 11-45

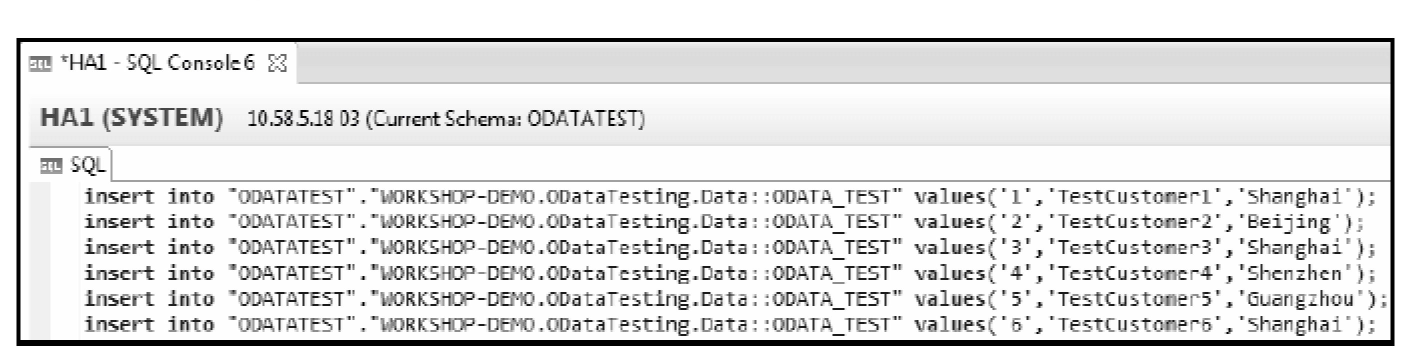
(6) 双击数据表进入数据表定义窗口，在此窗口中查看数据表结构和定义是否相符(见图 11-46)。



	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null
1	CUSTOMER_ID	INTEGER		INT	X(1)	X
2	CUSTOMER_NAME	NVARCHAR	60	STRING		X
3	CITY	NVARCHAR	20	STRING		

图 11-46

(7) 为数据表加入数据。为数据表加入数据有很多种方法可以实现，例如上载 CSV 数据文件，使用 SAP HANA 的数据提供(Data Provision)功能等。但是最快捷的办法就是使用 SQL 控制台直接插入数据，尤其是只需要少量的测试数据的时候。打开 SQL 控制台，输入如图 11-47 所示语句。




```
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('1','TestCustomer1','Shanghai');
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('2','TestCustomer2','Beijing');
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('3','TestCustomer3','Shanghai');
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('4','TestCustomer4','Shenzhen');
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('5','TestCustomer5','Guangzhou');
insert into "ODATATEST"."WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" values('6','TestCustomer6','Shanghai');
```

图 11-47

(8) 右击数据表，在右键菜单中选择“Open Content”(见图 11-48)。



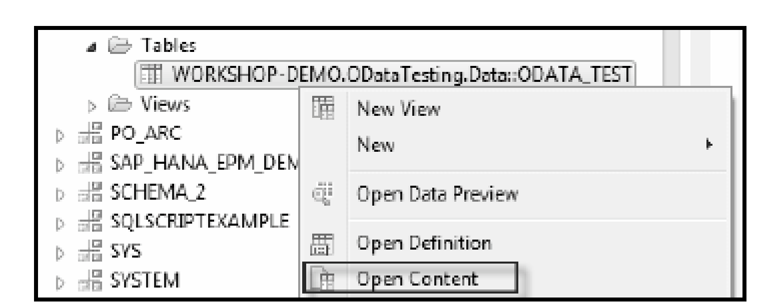


图 11-48

(9) 查看数据是否正确(见图 11-49)。

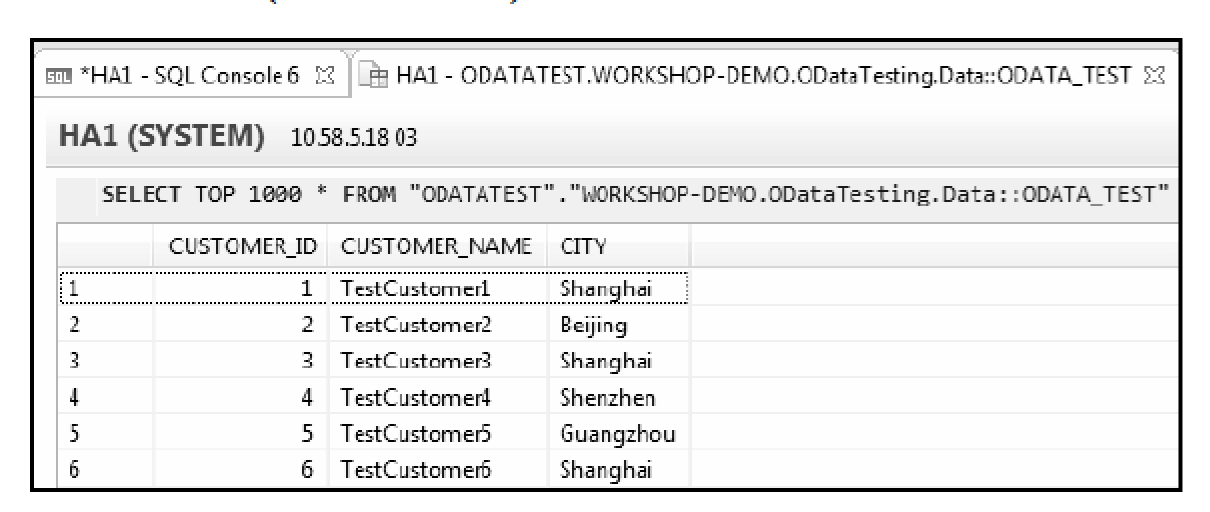


图 11-49

到目前为止，我们已经为 OData 服务准备好了需要暴露的数据源，接下来我们开始定义 OData 服务文件。

三、定义 OData 服务文件

(1) 在项目中创建“Service”文件夹。右击“ODataTesing”项目，选择“New”→“Folder”(见图 11-50)。

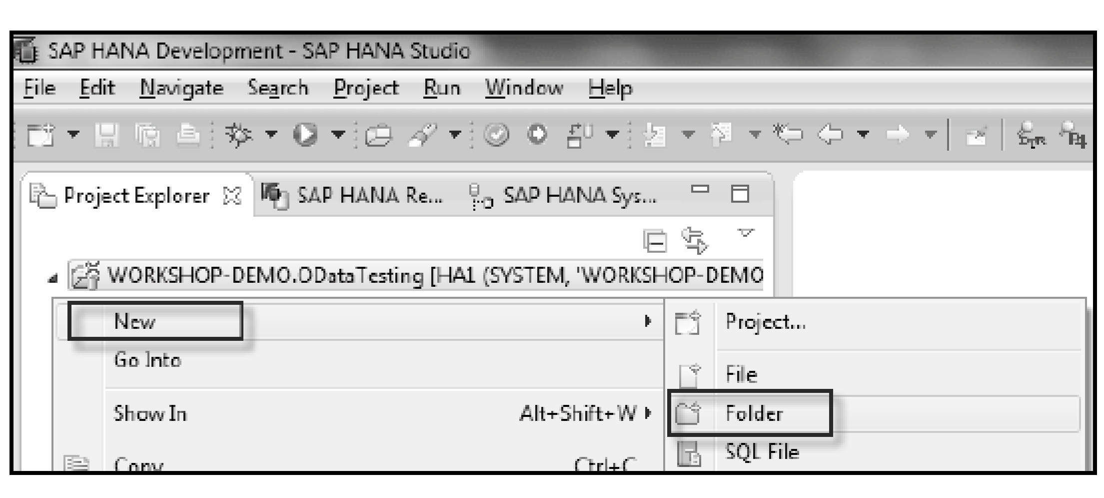


图 11-50



(2) 在弹出的“New Folder”窗口的“Folder name”字段中输入“Service”。单击“Finish”按钮确认(见图 11-51)。

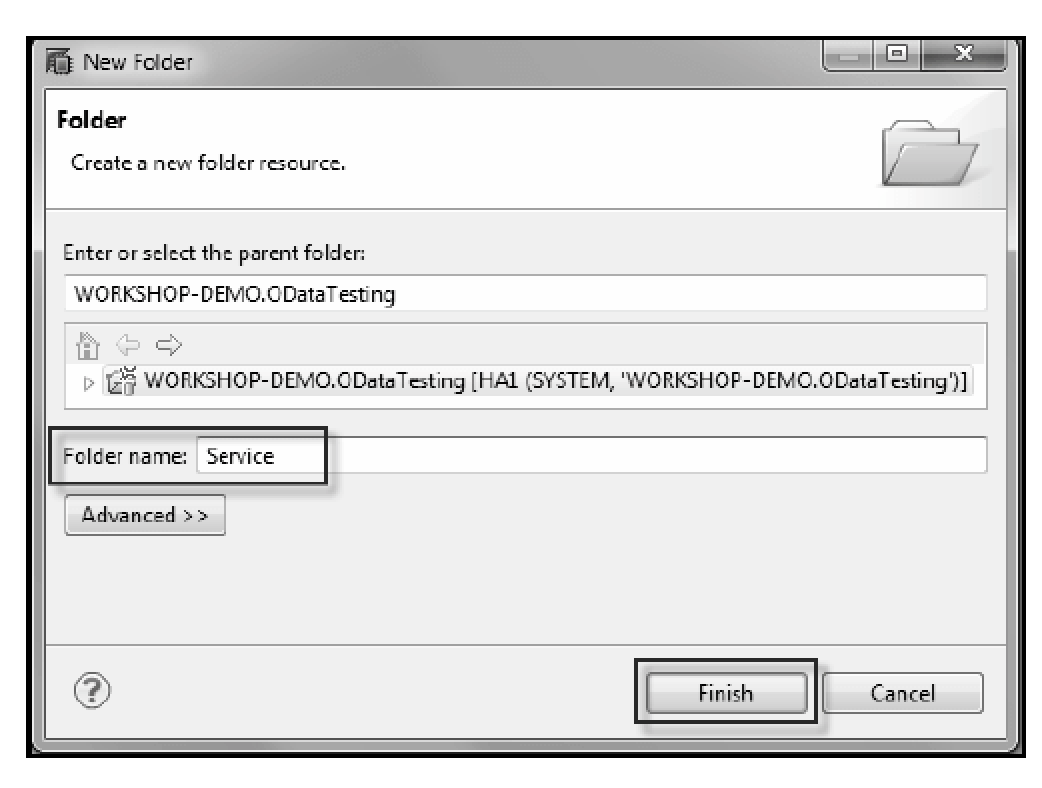


图 11-51

(3) 提交并激活新的“Service”文件夹(见图 11-52)。

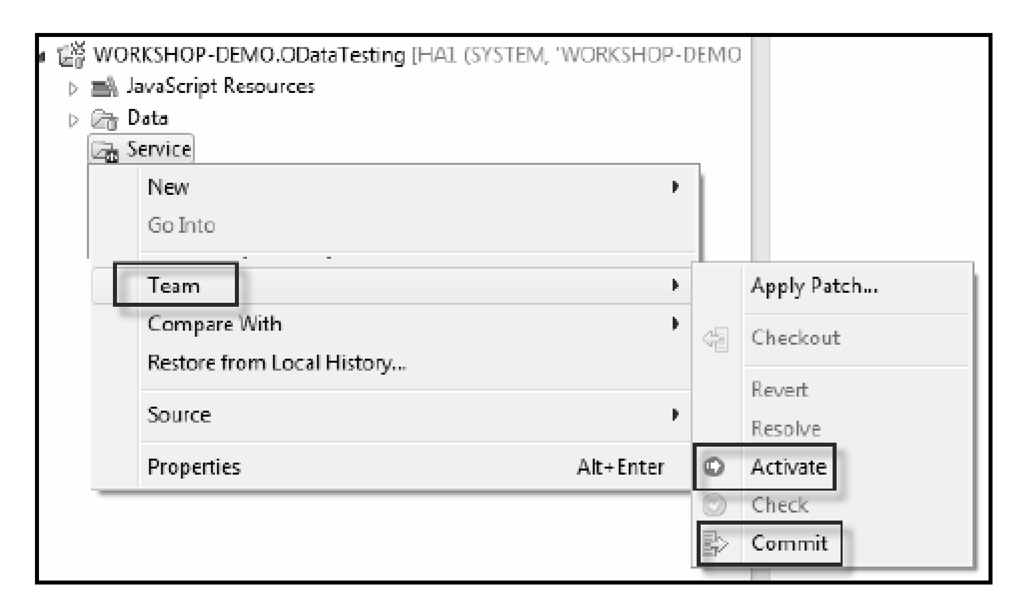


图 11-52

(4) 右击激活后的“Service”文件夹，在弹出的右键菜单中选择“New”→“File”(见图 11-53)。

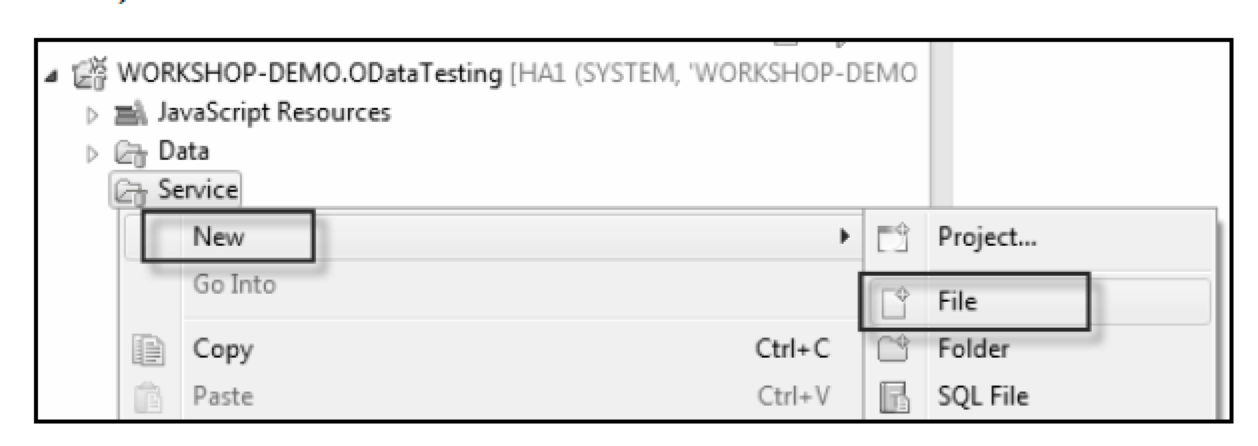


图 11-53

(5) 在弹出窗口中的“File name”字段输入“OdataSV.xsodata”，单击“Finish”按钮确认。OData 服务的定义文件是以“.xsodata”为后缀名的(见图 11-54)。



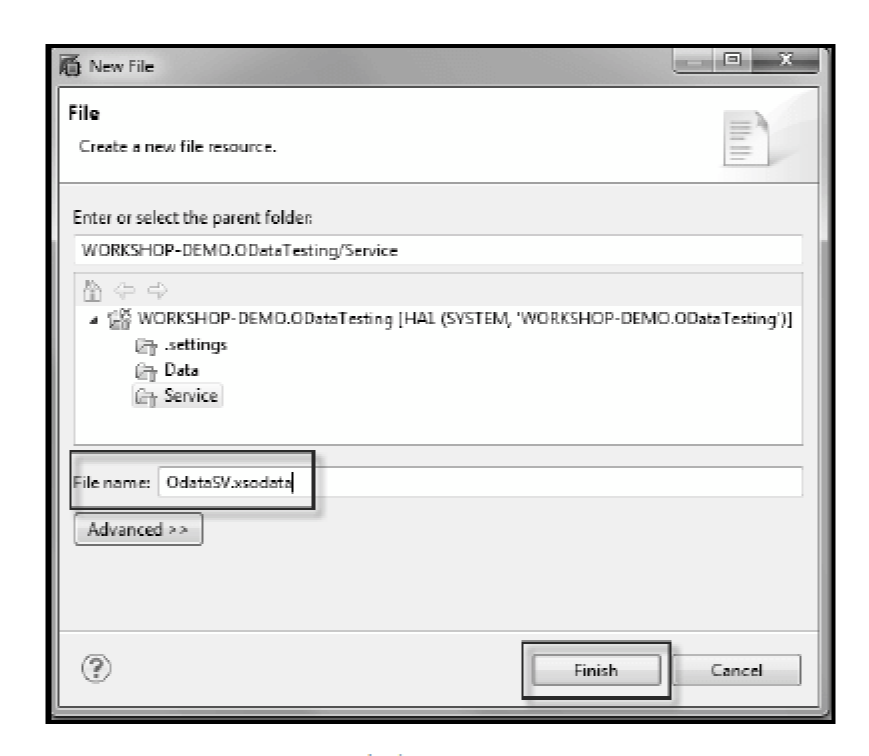



图 11-54

(6) 我们首先定义一个空的服务文件来测试 OData 服务。空 OData 服务的定义如图 11-55 所示，只需要在 OData 的定义文件中输入“service {}”即可。

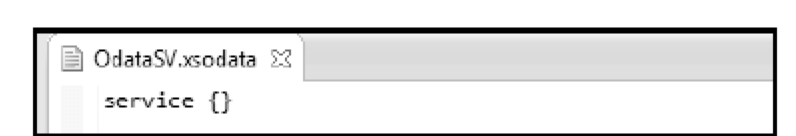


图 11-55

空 OData 服务非常有用，因为你不需要指定数据源，因此它可以非常方便地用来测试系统的 XS 服务以及 OData 服务是否正常而不用费力去寻找是否是数据源出了问题。

(7) 检查你的 XS 服务是否打开。通常来讲，检查 XS 服务的方法是在浏览器中输入“http://你的 HANA 服务器主机地址：8000”。“8000”这个数字的由来是组合“80”和你的服务器实例 ID。例如我的服务器地址为“10.58.5.18”，实例 ID 为“03”，那我就可以通过“http://10.58.5.18:8003”这个 URL 来测试我的 XS 服务。将“http://10.58.5.18:8003”输入浏览器中，如果 XS 服务正常会显示如图 11-56 所示的界面。





图 11-56

(8) 添加“.xsapp”和“.xsaccess”文件。我们先来添加“.xsapp”文件，这个文件是用来表明你项目中的所有对象都可以通过 HTTP 协议暴露。“.xsapp”文件没有内容，没有文件名，仅含有“.xsapp”后缀名，它在项目中的作用仅作为标示符存在。同样的，“.xsaccess”文件也仅含有后缀名，它与“.xsapp”文件的不同是它是定义控制权限的，因此我们需要为它添加内容(见图 11-57)。

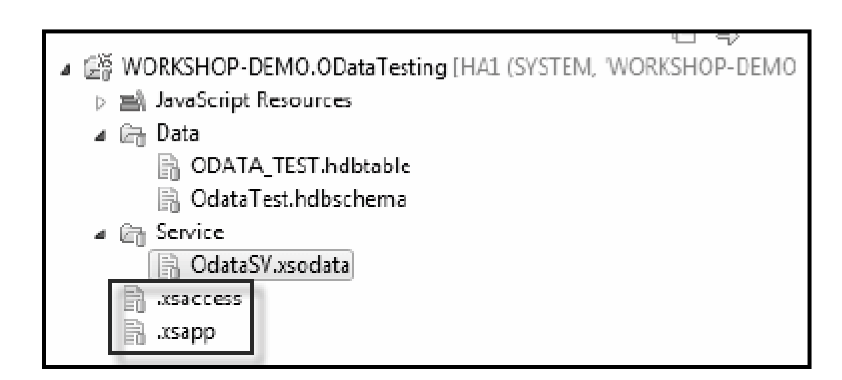


图 11-57

(9) 右击“ODataTesing”项目，选择“New”→“File”(见图 11-58)。

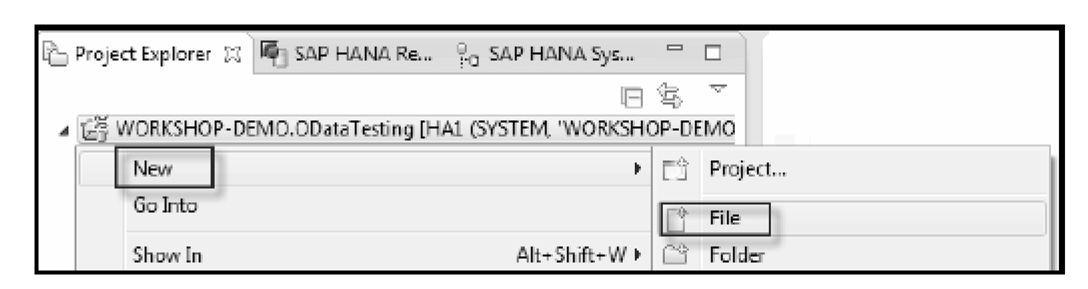


图 11-58

(10) 在弹出窗口的“File name”字段中输入“.xsapp”，单击“Finish”按钮确认(见图 11-59)。

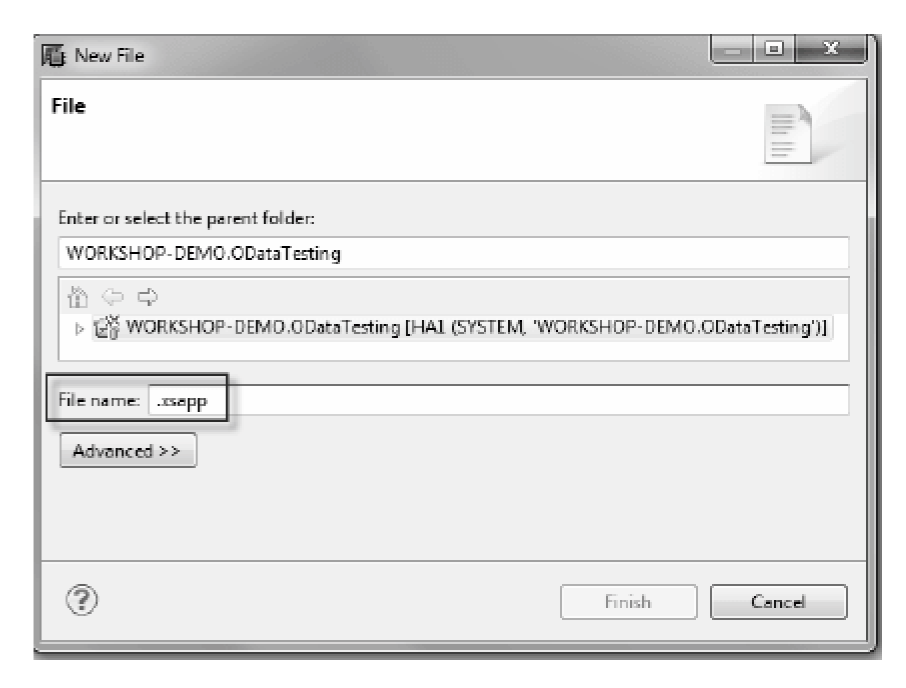


图 11-59

(11) 由于“.xsapp”文件是空的，因此我们不对它进行任何操作。接下来我们按照同样的方法创建“.xsaccess”文件(见图 11-60)。



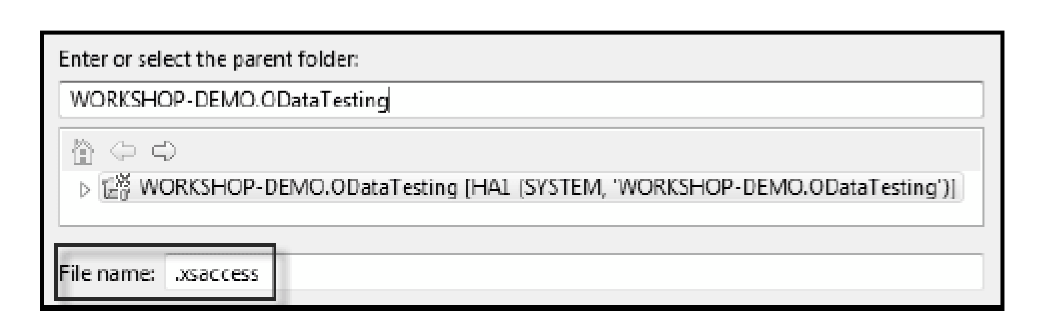


图 11-60

(12) 在编辑区将如下代码写入到“.xsaccess”文件中。代码包含两部分内容，一是指定对象是否可以暴露，二是指定对于权限控制的方法，本例中为“Basic”，“Basic”方式为使用通常的用户/密码方式访问 OData 服务(见图 11-61)。

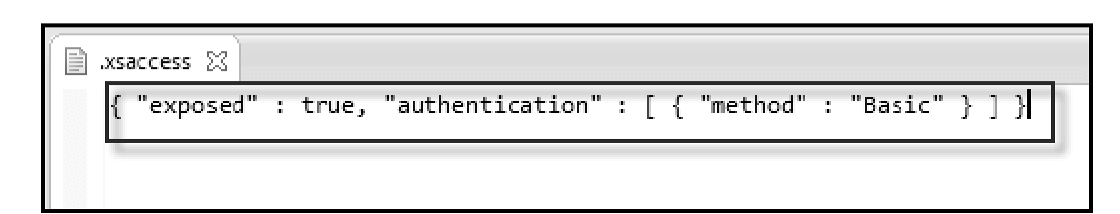


图 11-61

(13) 保存“.xsaccess”文件，然后将“.xsapp”文件和“.xsaccess”文件提交并激活。激活完成后我们就可以通过浏览器查看我们刚刚定义的 OData 服务文件了。调用 OData 服务的 URL 为 http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata。

先解释下 URL 的组成，“http://xxxxx”是 XS 服务的根目录，也就是我们刚刚测试 SAP HANA XS 服务的地址，“WORKSHOP-DEMO”是项目所在的包，“ODataTesting”是项目名称，“Service”是项目下文件夹名称，“OdataSV.xsodata”是 OData 服务的定义文件。将这些元素组合起来就变成调用 OData 服务的 URL (见图 11-62)。



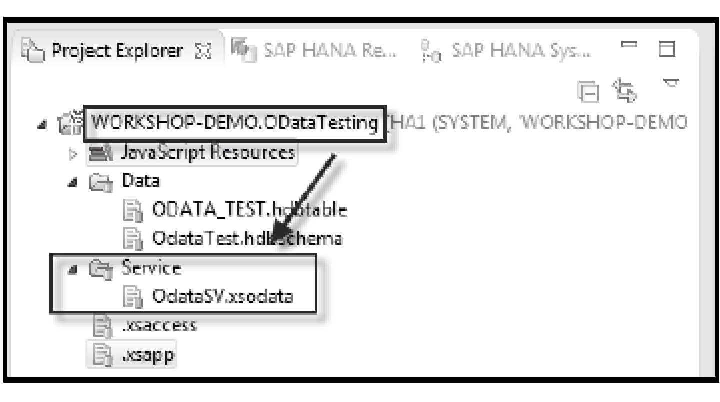


图 11-62

(14) 打开浏览器并在地址栏输入上述 URL。

在这里我们推荐使用“Google Chrome”浏览器，原因是“Google Chrome”浏览器可以方便地进行插件的拓展。这样我们能更加直观地检查数据。在我们今后的开发工作中，有两个“Google Chrome”插件是非常有用的，它们是“XML Tree”

和“JSONView”。请在你的“Google Chrome”浏览器中添加这两个插件的最新版本以方便我们日后的数据检查工作。除此以外，“Google Chrome”的开发者工具也是非常有用的。

(15) 输入 URL 并回车后，系统会要求你输入用户名和密码，这是“.xsaccess”文件起作用的表现，因为我们在创建“.xsaccess”文件时选择了“Basic”方式，所以系统要求我们输入用户名/密码。请在图 11-63 所示对话框中输入你登录 SAP HANA Studio 的用户信息。

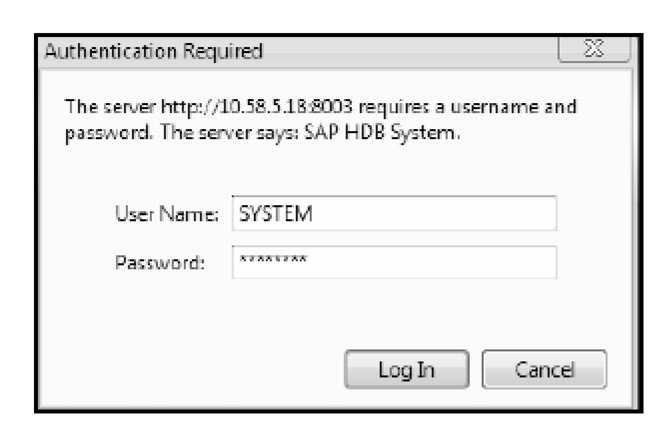


图 11-63

(16) 输入完成后单击“Log In”按钮，如果出现图 11-64 显示的界面，说明我们调用 OData 服务成功。图 11-64 的界面是“Google Chrome”通过 XML Tree 插件形成的界面，如果你没有安装此插件，则界面会是标准的 XML 显示界面。



图 11-64

(17) 接下来我们看看 OData 服务的元数据，在浏览器中输入“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/$metadata”(见图 11-65)。



图 11-65

当我们需要调用 OData 服务的元数据时，我们需要在 URL 末尾加“$metadata”参数。对于图 11-65 的返回结果有两个地方需要说明一下。

- 首先是“Namespace”。从返回结果中我们可以看到系统自动为我们的 OData 服务指定了默认的命名空间：<Namespace="WORKSHOP-DEMO.ODataTesting.Service.OdataSV">，这是由于我们在当初定义 OData 服务时没有指定命名空间。

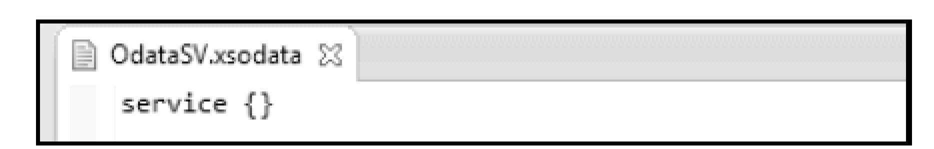


图 11-66

如果你想要使用自己的命名空间，那么你需要在定义 OData 服务时就在代码中明确标明出来，如图 11-67 所示。



图 11-67

刷新刚才的页面后你就能在元数据页面看到刚才定义的命名空间(见图 11-68)。

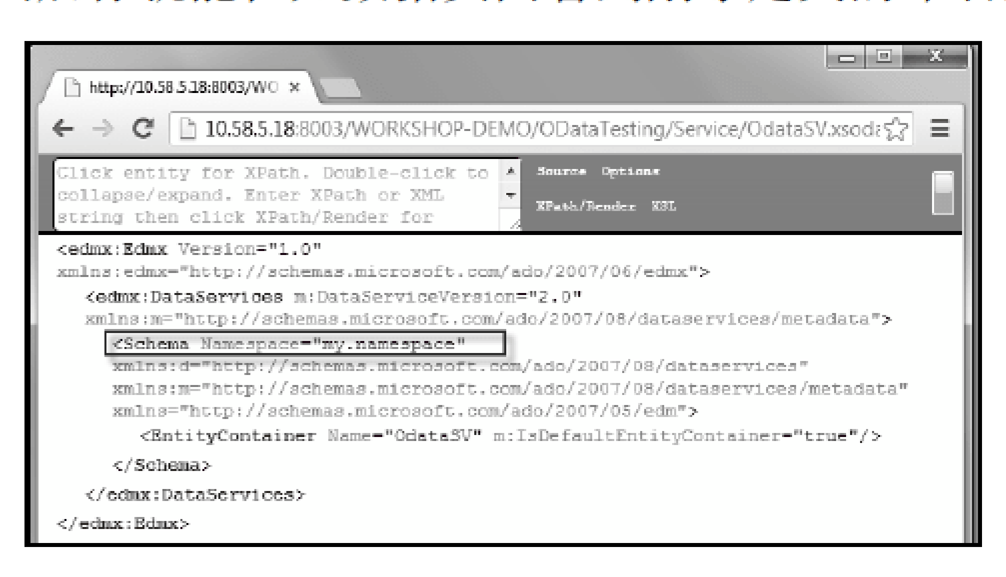


图 11-68



- “Entity Container”：数据实体容器，名称与 OData 服务定义文件名称一致。我们今后真正暴露数据源时，所有的数据实体都会在这个容器中体现。

调用成功后，接下来我们就需要真正暴露我们的测试数据表来看看 OData 服务通过 SAP HANA XS 调用的数据是怎么反映到浏览器的。

四、创建简单的 OData 服务

通常来讲，我们可以在 OData 服务中暴露任何 SAP HANA Studio 中的数据源。

这些数据源包括数据表、SQL 视图、属性视图、分析视图和计算视图等。我们在本小节中先定义一个基于数据表的简单 OData 服务，这种 OData 服务是我们在今后的开发工作中最经常用到的数据传输服务之一。

(1) 打开 SAP HANA Studio，进入“SAP HANA Development”透视图，打开“Project Explorer”视图(见图 11-69)。

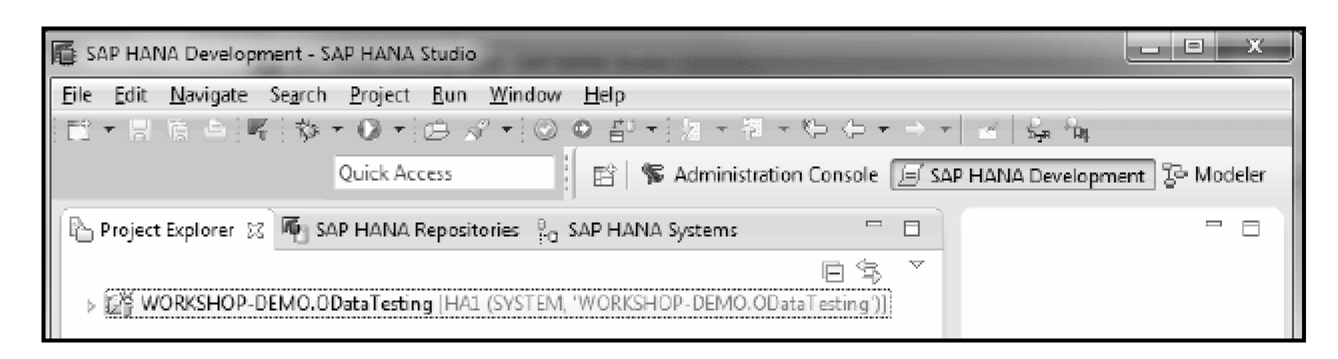


图 11-69

(2) 展开“WORKSHOP-DEMO.ODataTesing”项目，定位到“Service”→“OdataSV.xsodata”文件，即我们在上一小节生成的 OData 服务定义文件。双击“OdataSV.xsodata”文件，在右侧编辑区域内输入如下代码：

```
service namespace "odata.test"{"WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST" as "My.Test.Table";}
```

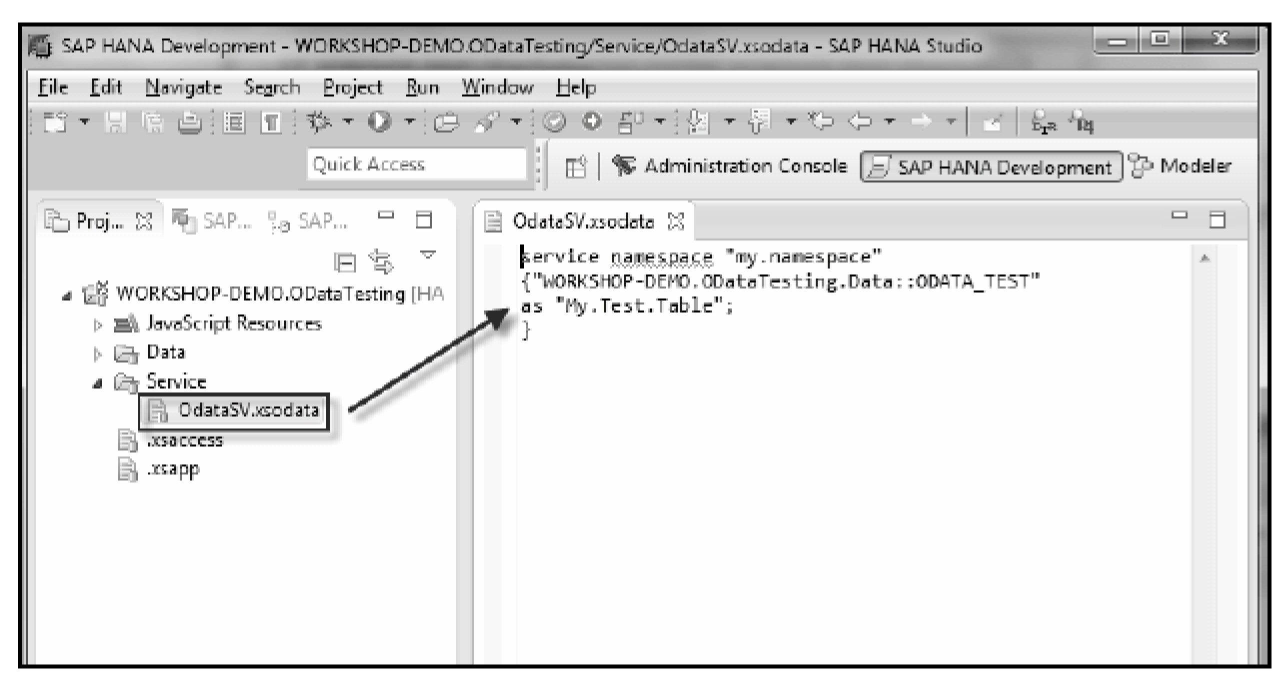


图 11-70

这段代码的含义很简单，我们在这个 OData 服务中定义了“odata.test”作为命名空间以表明我们不会使用系统默认的命名空间。在大括号里面我们首先直接引用了项目“WORKSHOP-DEMO.ODataTesing”设计时(Design-Time)的数据表“WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST”作为暴露数据源，但是由于原来的数据表含有很长的命名空间“WORKSHOP-DEMO.ODataTesting.Data::”，因此我们使用“as”参数来自定义数据源的显示名称，这里我们定义输出

的表名为“My.Test.Table”。输入完成后，提交并激活“OdataSV.xsodata”文件。

(3) 在浏览器中调用“OdataSV.xsodata”服务元数据(OData 服务 URL 后面加“$medatada”参数)。URL 为“http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/$metadata”。界面如图 11-71 所示。


```
<edmx:Edmx Version="1.0" xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
  <edmx:DataServices m:DataServiceVersion="2.0"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    <Schema Namespace="odata.test" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns="http://schemas.microsoft.com/ado/2007/05/edm">
      <EntityType Name="My.Test.TableType">
        <Key>
          <PropertyRef Name="CUSTOMER_ID"/>
        </Key>
        <Property Name="CUSTOMER_ID" Type="Edm.Int32" Nullable="false"/>
        <Property Name="CUSTOMER_NAME" Type="Edm.String" Nullable="false" MaxLength="60"/>
        <Property Name="CITY" Type="Edm.String" Nullable="true" MaxLength="30"/>
      </EntityType>
      <EntityContainer Name="OdataSV" m:IsDefaultEntityContainer="true">
        <EntitySet Name="My.Test.Table" EntityType="odata.test.My.Test.TableType"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```


图 11-71

在图 11-71 元数据浏览界面，我们可看到“WORKSHOP-DEMO.ODataTesting.Data::ODATA_TEST”数据表在输出时被重命名为“My.Test.Table”，但是其在 SAP HNA Studio 里面所有的结构定义信息都被显示出来了，例如该数据表包含的三个字段，每个字段的具体设置以及哪个字段是关键字段等信息。

(4) 调用“My.Test.Table”数据表内容(OData 服务 URL 后面加“重命名表”名称)“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table/”。界面如图 11-72 所示。


```
<feed xml:base="http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/" xmlns:d="http://schemas.microsof
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom">
  <title type="text">My.Test.Table</title>
  <id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table</id>
  <author>
    <name/>
  </author>
  <link rel="self" title="My.Test.Table" href="My.Test.Table"/>
  <entry>
    <id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)</id>
    <title type="text"/>
    <author>
      <name/>
    </author>
    <category term="my.namespace.My.Test.TableType" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
    <content type="application/xml">
      <m:properties>
        <d:CUSTOMER_ID m:type="Edm.Int32">1</d:CUSTOMER_ID>
        <d:CUSTOMER_NAME m:type="Edm.String">TestCustomer1</d:CUSTOMER_NAME>
        <d:CITY m:type="Edm.String">Shanghai</d:CITY>
      </m:properties>
    </content>
  </entry>
```


图 11-72



首先我们能在输出页面看到我们需要暴露的数据表中的所有数据都显示了出来(由于页面的限制我们只给出第一条数据的截图)，然后我们研究下上图蓝框内标出的 URL 究竟代表什么意义，将此 URL 输入浏览器“http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)”。界面如图 11-73 所示。

```
<entry xml:base="http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns="http://www.w3.org/2005/Atom">
  <id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)</id>
  <title type="text"/>
  <author>
    <name/>
  </author>
  <category term="my.namespace.My.Test.TableType"
  scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
  <content type="application/xml">
    <m:properties>
      <d:CUSTOMER_ID m:type="Edm.Int32">1</d:CUSTOMER_ID>
      <d:CUSTOMER_NAME m:type="Edm.String">TestCustomer1</d:CUSTOMER_NAME>
      <d:CITY m:type="Edm.String">Shanghai</d:CITY>
    </m:properties>
  </content>
</entry>
```

图 11-73

在输出界面中你会发现只有关键字“CUSTOMER_ID”=“1”的记录被显示出来。在 OData 服务中，数据源的每一条数据都有属于自己的 URL，这个 URL 以数据表的关键字来做参数，因此每个 URL 都是唯一的。这种记录专属的 URL 对于我们今后开发中需要针对某条记录进行操作或者根据某条记录来进行界面导航时有非常大的作用。在这个例子中，我们通过指定的 URL 除了可以单独访问数据表的某条记录以外，还可以通过如下的请求单独暴露某条记录的某个字段，甚至只暴露该字段的内容而不包含其属性，具体例子如下。

- 暴露某条记录的某个字段。还是第一条记录为例，我们在刚才的 URL 后面加上需要暴露的字段，例如我们只暴露所在城市“CITY”字段“http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)/CITY”。结果如图 11-74 所示。

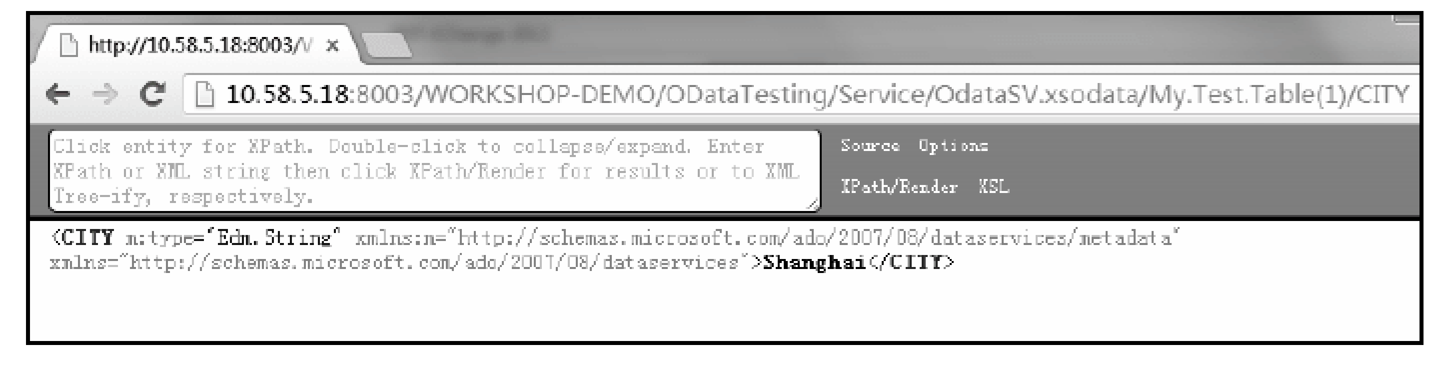


图 11-74



- 有时候，我们不需要将字段的属性全部暴露出来，如果我们只想暴露字段的数值，那么我们可以在使用“$value”参数来定义只暴露字段的数值，例如我们接着上面的例子只暴露所在城市“CITY”字段的数值“http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)/CITY/$value”。结果如图 11-75 所示。

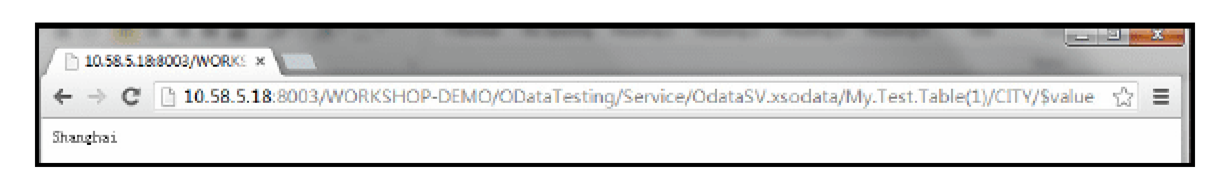


图 11-75

(5) 转换 OData 服务的输出格式。SAP HANA XS 中的 OData 服务默认是以“Atom”格式输出内容的。Atom 供稿格式(Atom Syndication Format)是用于网站消息来源基于 XML 的文档格式，而 Atom 出版协定(Atom Publishing Protocol，简称 AtomPub 或 APP)是用于新增及修改网络资源的一种基于 HTTP 的协议。由于“Atom”是基于 XML 的文档格式，因此它对于不包含 JavaScript 的业务应用具有非常好的兼容性和易用性，尤其是微软的开发工具，如“Microsoft Visual Studio”等能够非常好的支持“Atom”格式的文件。除此以外，OData 服务还提供“JSON”格式的输出文件。JSON(JavaScript Object Notation)是一种轻量级的数据交换格式。它使得包含有 JavaScript 的应用程序可以非常快的解析该数据文件所包含的数据，相比于 XML 格式，JSON 在包含有 JavaScript 的程序中能比 XML 文件更快地被解析，因此程序的执行性能会提高很多。JSON 数据格式大部分应用在包含有 JavaScript 的 Web 程序中。



下面我们看看把 OData 服务的输出格式由默认的“Atom”格式转换为“JSON”格式的 URL(见图 11-76)，所需的 URL 为：“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$format=json”。


```
{
 - d: {
    - results: [
       + {…},
       + {…},
       + {…},
       - {
          - __metadata: {
               uri: "http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(4)",
               type: "odata.test.My.Test.TableType"
            },
            CUSTOMER_ID: 4,
            CUSTOMER_NAME: "TestCustomer4",
            CITY: "Shenzhen"
         },
```


图 11-76

在这个 URL 中，我们首先把输出定位到需暴露的数据表，然后我们增加"$format=json"参数把输出格式转换成 JSON 格式。除了转换格式以外，我们还可以在 OData 服务中添加对于记录进行查询的参数。在下一小节中，我们将详细介绍 OData 服务中数据请求的具体内容。

五、OData 服务数据请求——查询

在 OData 服务中，数据请求(Data Request)是其中一个重要组成部分。像我们之前介绍的元数据请求(Metadata Request)以及单条数据请求(Individual Entities Request)等都包含在数据请求服务中。为了让大家对于 OData 服务的数据请求有一个更好的概念，我们在这一小节将会着重介绍 OData 服务的数据请求查询相关参数，以方便大家在今后的开发工作中使用。

- "$top"参数。"$top"参数是用来定义只有最开始的 N 条记录才被返回，例如"$top=2"代表只返回最初的两条记录。请试着将包含"$top"参数的 URL"http://xxxxxxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$top=2"在浏览器中打开并检查得到的结果。
- "$skip"参数。"$skip"参数是用来定义最开始的 N 条记录一定不被返回，例如"$skip=2"代表最初的两条记录不能被返回，其余记录可以被返回。请试着将包含"$skip"参数的 URL "http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$skip=2" 在浏览器中打开并检查得到的结果。
- "$filter"参数。"$filter"参数是起到根据给出的条件限制输出数据的作用的。"$filter"参数需要与布尔表达式一起使用来达到数据输出限制的目的。"$filter"参数具体用法请参照表 11-1。

表 11-1

操作	描述	例子
逻辑操作		
eq	等于	/Suppliers?$filter=City eq 'Shanghai'
ne	不等于	/Suppliers?$filter=City ne 'Shenzhen'
gt	大于	/Products?$filter=Price gt 30
ge	大于等于	/Products?$filter=Price ge 20
lt	小于	/Products?$filter=Price lt 30

(续表)

操作	描述	例子
le	小于等于	/Products?$filter=Price le 1000
and	逻辑与	/Products?$filter=Price le 200 and Price gt 3.5
or	逻辑或	/Products?$filter=Price le 3.5 or Price gt 200
not	逻辑非	/Products?$filter=not endswith(Description,'Notebook')
运算操作		
add	加法	/Products?$filter=Price add 5 gt 10
sub	减法	/Products?$filter=Price sub 5 gt 10
mul	乘法	/Products?$filter=Price mul 2 gt 2000
div	除法	/Products?$filter=Price div 2 gt 4
mod	求余	/Products?$filter=Price mod 2 eq 0
组运算		
( )	优先组	/Products?$filter=(Price sub 5) gt 10

- “$select”参数。“$select”参数是用来定义哪些符合条件的属性值(Properties)能被返回，例如“$select= CITY”代表只有具有“CITY”属性的值才被返回。请试着将包含“$select”参数的 URL “http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$select=CITY”在浏览器中打开并检查得到的结果。为了更直观地描述此参数，我们可以看图 11-77 所示的输出截屏。



图 11-77



从图 11-77 中我们可以看出只有属性为“CITY”的输出字段被显示出来，而其他字段都被隐含。因此这个参数在控制 OData 服务输出列方面非常有用。我们在图 11-78 中对比一下以前不加“$select= CITY”参数的结果，可以看出所有字段(数据列)都被输出。

```
<id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table(1)</id>
<title type="text"/>
<author>
  <name/>
</author>
<category term="odata.test.My.Test.TableType" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<content type="application/xml">
  <m:properties>
    <d:CUSTOMER_ID m:type="Edm.Int32">1</d:CUSTOMER_ID>
    <d:CUSTOMER_NAME m:type="Edm.String">TestCustomer1</d:CUSTOMER_NAME>
    <d:CITY m:type="Edm.String">Shanghai</d:CITY>
  </m:properties>
</content>
```

图 11-78

有一点需要说明的是，“$select”参数只对于 OData 服务使用“Atom”格式输出文件中的“<m:properties></m:properties>”节点内的内容生效，对于其他节点的内容是无效的。

- “$orderby”参数。“$orderby”参数是用来将输出数据按照指定的某一字段进行排序的参数，请试着将包含“$orderby”参数的 URL “http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$orderby=CITY asc”在浏览器中打开并检查得到的结果。在“$orderby”参数中，如果你指定“asc”为后缀则说明按照升序排列，如果你指定“desc”则说明按照降序排列。同时你还可以针对两个列进行分别排序，例如：http://services.odata.org/OData/OData.xsdata/Products?$orderby=ReleaseDate asc, Rating desc。
- “$inlinecount”参数。“$inlinecount”参数是为 OData 服务返回值提供一种临时的用来统计条目数的参数。如果你在 OData 服务的请求 URL 里面使用此参数，则能得到此次请求的所有数据条目数。请试着将包含“$inlinecount”参数的 URL“http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/OdataSV.xsodata/My.Test.Table?$ inlinecount=allpages”在浏览器中打开并检查得到的结果(见图 11-79)。



图 11-79

有一点需要指出的是，这些参数不仅能够单独使用，还能组合在一起使用。举个常见的例子，如果我们想要在用户界面做到自动分页的效果，比如每 40 条记录为一页，我们可以使用“$top”和“$skip”参数的组合来实现这一功能。具体代码如下(只包含前三页)：

页一：http://<HANAhost>:80<instance>/demo.xsodata?$top=40&$skip=0

页二：http://<HANAhost>:80<instance>/demo.xsodata?$top=40&$skip=40

页三：http://<HANAhost>:80<instance>/demo.xsodata?$top=40&$skip=80

到此为止，我们已经介绍了如何创建简单的基于数据表的 OData 服务，以及如何在 OData 服务中调用数据请求和增加查询参数。接下来，让我们看看如何在 SAP HANA Studio 中创建更加复杂的 OData 服务，例如多个表的数据暴露，或者是属性视图、分析视图以及计算视图的数据暴露等。



第三节　创建复杂的 OData 服务

在上一小节中，我们创建了一个非常简单的 OData 服务来熟悉在 SAP HANA XS 服务器怎么调用 OData 服务以及如何进行数据请求。在本小节中，我们将介绍一些更复杂的 OData 服务的创建来为大家今后的开发工作作参考。

一、为 OData 服务定义属性映射

在 SAP HANA Studio 中，为 OData 服务定义属性映射(Property Projection)的概念有点类似于上面讲的在 OData 数据请求 URL 中加“$select”参数的概念，即限制 OData 服务暴露的数据源的列数。但是属性映射不同于在数据请求 URL 中增加“$select”

参数，它是在定义 OData 服务的时候就在 OData 定义文件里面明确指出需要输出的列是哪些，而且由于属性映射支持包含于(including)和排除于(Excluding)两种方法来定义输出列，因此比“$select”参数具有更大的灵活性。接下来我们详细看看如何定义 OData 服务的属性映射。

(1) 首先我们要使用一个复杂的，含有多个列的数据表作为需暴露的数据源，在这里我们将使用“SAP_HANA_EPM_DEMO”节点中的“sap.hana.democontent.epm.data::purchaseOrder”表作为测试表(见图 11-80)。

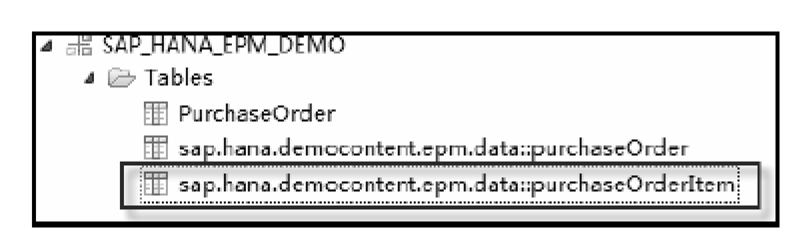


图 11-80

(2) 该数据表具有如图 11-81 所示的输出列。

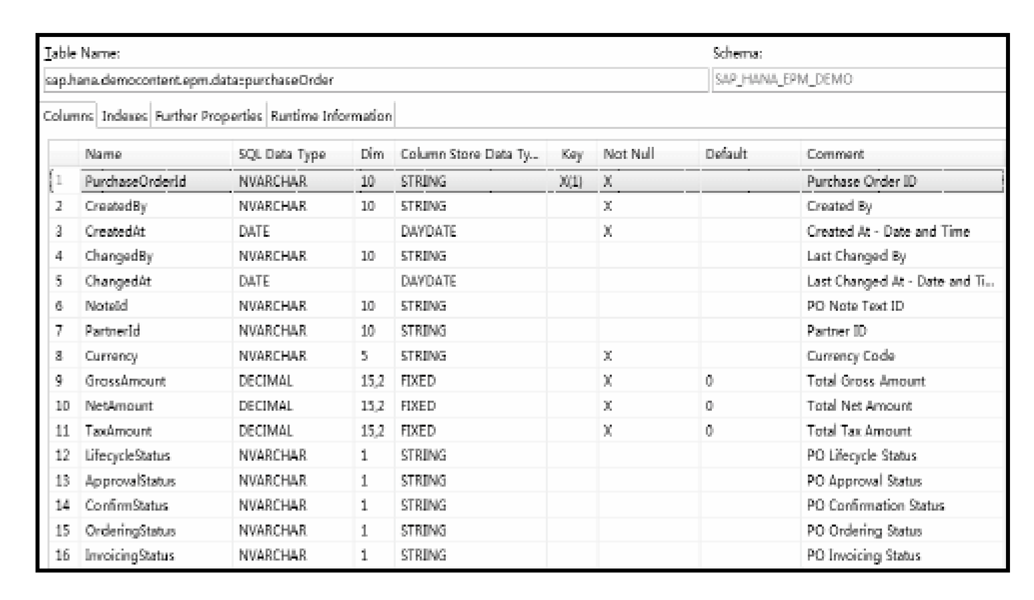


Table Name: sap.hana.democontent.epm.data::purchaseOrder

Schema: SAP_HANA_EPM_DEMO

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Ty...	Key	Not Null	Default	Comment
1	PurchaseOrderId	NVARCHAR	10	STRING	X(1)	X		Purchase Order ID
2	CreatedBy	NVARCHAR	10	STRING		X		Created By
3	CreatedAt	DATE		DAYDATE		X		Created At - Date and Time
4	ChangedBy	NVARCHAR	10	STRING				Last Changed By
5	ChangedAt	DATE		DAYDATE				Last Changed At - Date and Ti...
6	NoteId	NVARCHAR	10	STRING				PO Note Text ID
7	PartnerId	NVARCHAR	10	STRING				Partner ID
8	Currency	NVARCHAR	5	STRING		X		Currency Code
9	GrossAmount	DECIMAL	15,2	FIXED		X	0	Total Gross Amount
10	NetAmount	DECIMAL	15,2	FIXED		X	0	Total Net Amount
11	TaxAmount	DECIMAL	15,2	FIXED		X	0	Total Tax Amount
12	LifecycleStatus	NVARCHAR	1	STRING				PO Lifecycle Status
13	ApprovalStatus	NVARCHAR	1	STRING				PO Approval Status
14	ConfirmStatus	NVARCHAR	1	STRING				PO Confirmation Status
15	OrderingStatus	NVARCHAR	1	STRING				PO Ordering Status
16	InvoicingStatus	NVARCHAR	1	STRING				PO Invoicing Status

图 11-81

(3) 接下来我们在项目中创建一个新的 OData 服务定义文件“odata_pro_proj_t.xsodata”(见图 11-82)。

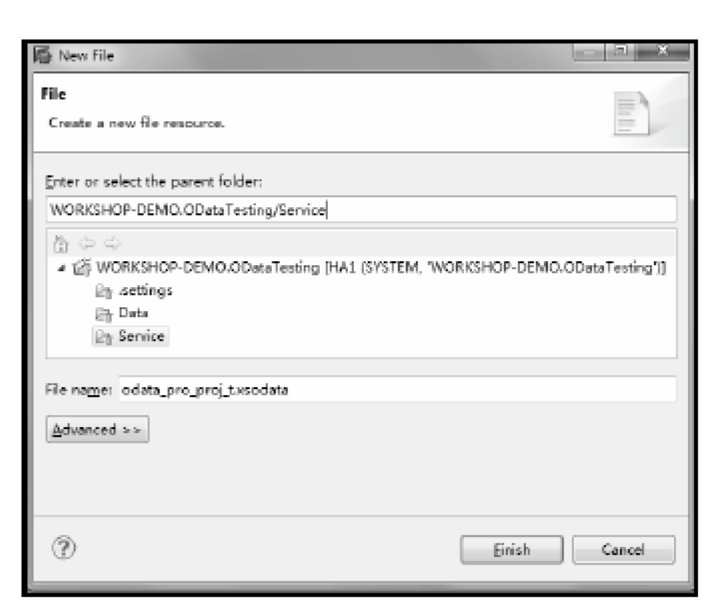


图 11-82

(4) 在编辑区，我们为新的 OData 服务定义属性映射代码，如图 11-83 所示。

```
odata_pro_proj_t.xsodata
service namespace "odata.test"
{"SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::purchaseOrder" as "purchaseorder"
with ("PurchaseOrderId","ProductId","GrossAmount","NetAmount","Currency");
}
```

图 11-83

这段代码与我们最初的 OData 服务文件创建代码的不同就在于我们增加了“with”参数，在“with”参数中我们指定了“PurchaseOrderId”、“ProductId”、“GrossAmount”、“NetAmount”、“Currency”为输出列。将此 OData 服务文件提交并激活后，我们在浏览器中将它打开(见图 11-84)。

```
<EntityType Name="purchaseorderType">
  <Key>
    <PropertyRef Name="PurchaseOrderId"/>
  </Key>
  <Property Name="PurchaseOrderId" Type="Edm.String" Nullable="false" MaxLength="10"/>
  <Property Name="Currency" Type="Edm.String" Nullable="false" MaxLength="5"/>
  <Property Name="GrossAmount" Type="Edm.Decimal" Nullable="false" Precision="15" Scale="2"/>
  <Property Name="NetAmount" Type="Edm.Decimal" Nullable="false" Precision="15" Scale="2"/>
</EntityType>
```

图 11-84

在 OData 服务元数据中，所有我们在定义文件中指定的列被显示了出来。

(5) 将“with”参数替换成“without”参数。前面我们说过，属性映射除了提供包含于(including)方法(通过“with”参数实现)，还提供了排除于(excluding)方法(通过“without”参数实现)来达到限制输出列的作用。对于一个包含有许多数据列，而且只有几个列不需要输出的数据源来说，使用“without”参数能起到更快的效率。例如我们只屏蔽“NoteId”列，其他都作为输出列(见图 11-85)。




```
odata_pro_proj_t.xsodata
service namespace "odata.test"
{"SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::purchaseOrder" as "purchaseorder"
without ("NoteId");
}
```

图 11-85

将更改后的 OData 定义文件提交并激活后，我们在浏览器中将它打开(见图 11-86)。

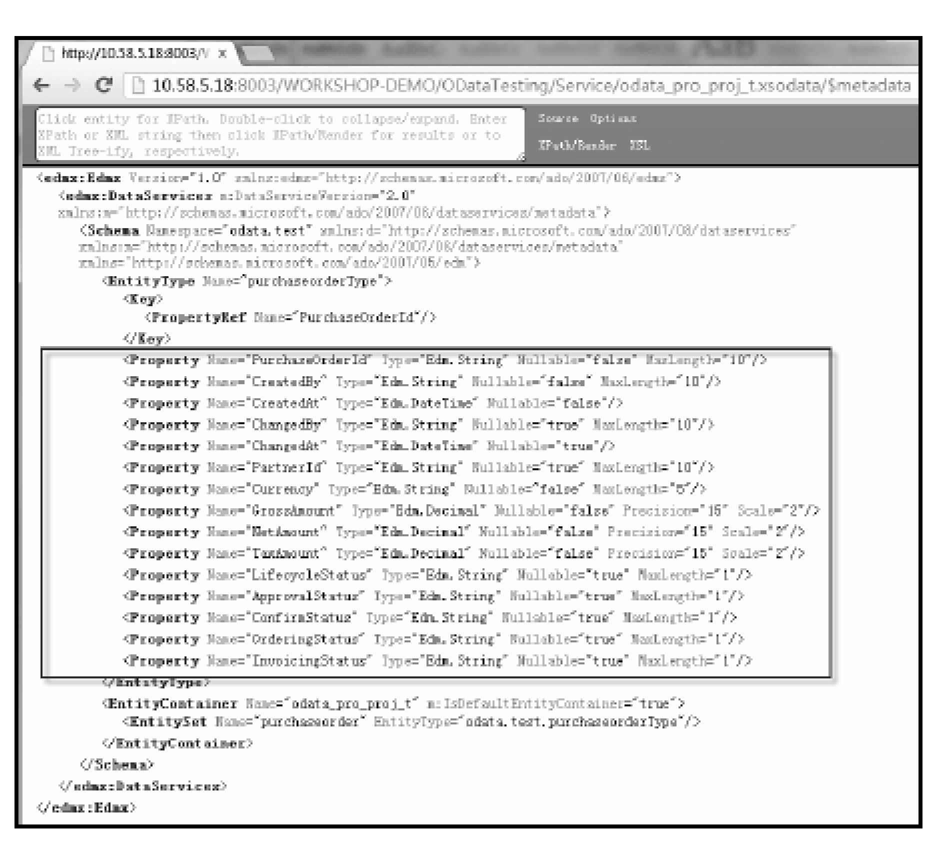


图 11-86

在 OData 服务元数据中，除了我们指定的“NoteId”列，其他所有的列都被显示了出来。

二、为 OData 服务定义关联

在 OData 服务的定义文件中，我们可以通过定义各个实体之间的关联(Associations)来表述各实体之间的联系。例如我们在 OData 定义文件中包含“采购订单”和“产品”两张表，为了更清楚地在输出结果中体现这两张表之间的关联，我们需要在 OData 定义文件中为这两张表定义关联，具体操作如下：

(1) 在项目中创建新的 OData 定义文件“odata_assoc_t.xsodata”(见图 11-87)。

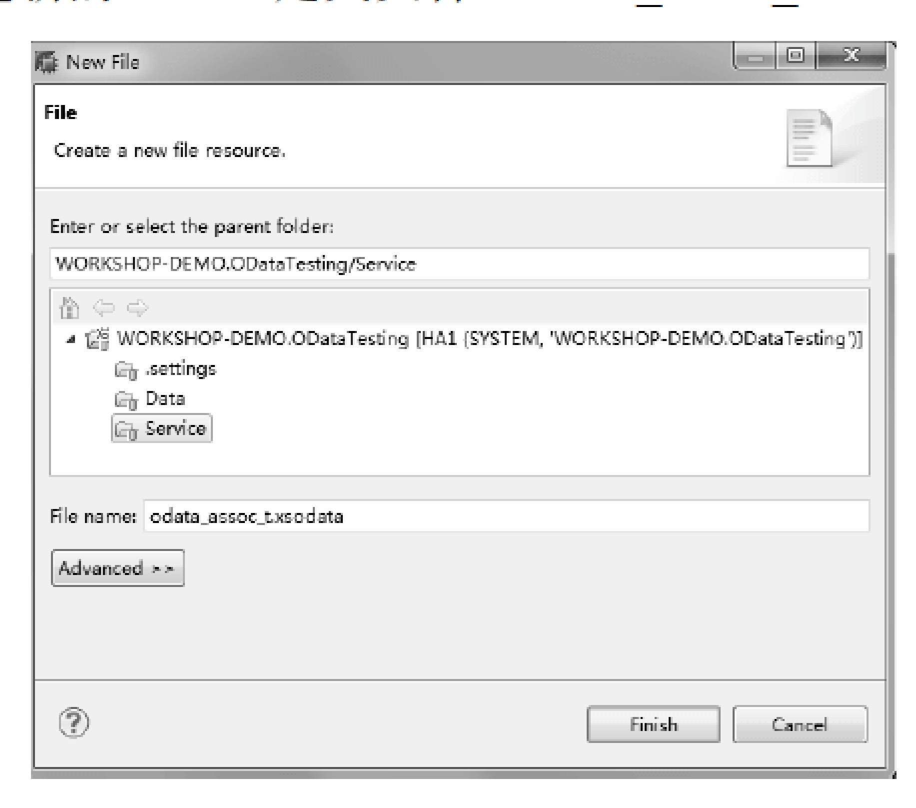


图 11-87



(2) 在编辑区输入如图 11-88 所示的代码。

```
*odata_assoc_t.xsodata
service namespace "odata.test"
{   "SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::purchaseOrderItem" as "purchaseorder"
    navigates ("purchase_product" as "npur_product");
    "SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::products" as "product";
     association "purchase_product" principal "purchaseorder"("ProductId") multiplicity "1"
     dependent "product"("ProductId") multiplicity "1";
}
```

图 11-88

这段代码稍微有点复杂，我们先来看看这段代码的结构：请注意在这段代码中出现的三个分号，它们的作用是起到分隔代码作用，既我们对每一个实体进行的定义操作都需要由分号分隔开。在这段代码中，我们分别定义了三个实体(红框标出)，它们是“purchaseorder”、“product”、“purchase_product”。

“purchaseorder”和“product”这两个实体分别对应我们引用的两个 SAP HANA 数据表，我们在前面的小节已经详细讲述了它们之间的对应关系。“purchase_product”实体对应的是“purchaseorder”和“product”这两个实体之间的关联，即“"purchaseorder"("ProductId")<1:1>"product"("ProductId")”。由于单独定义实体间的关联(association)是无法在 OData 服务中实现一个实体导航到另一个实体的功能的，所以我们必须在 OData 服务定义文件中加入导航属性(Navigation Properties)，在上图中黑框标出的语句即为定义导航属性的语句，通过该语句我们定义了一个名为“npur_product”的导航属性，并且标明“purchaseorder”实体可以通过“purchase_product”关联进行导航。下面我们来通过在浏览器中的实际输出结果来对这个 OData 服务进行更深入的理解。



(3) 调用“odata_assoc_t.xsodata”OData 服务的元数据：“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/odata_assoc_t.xsodata/$metadata”(见图 11-89)。

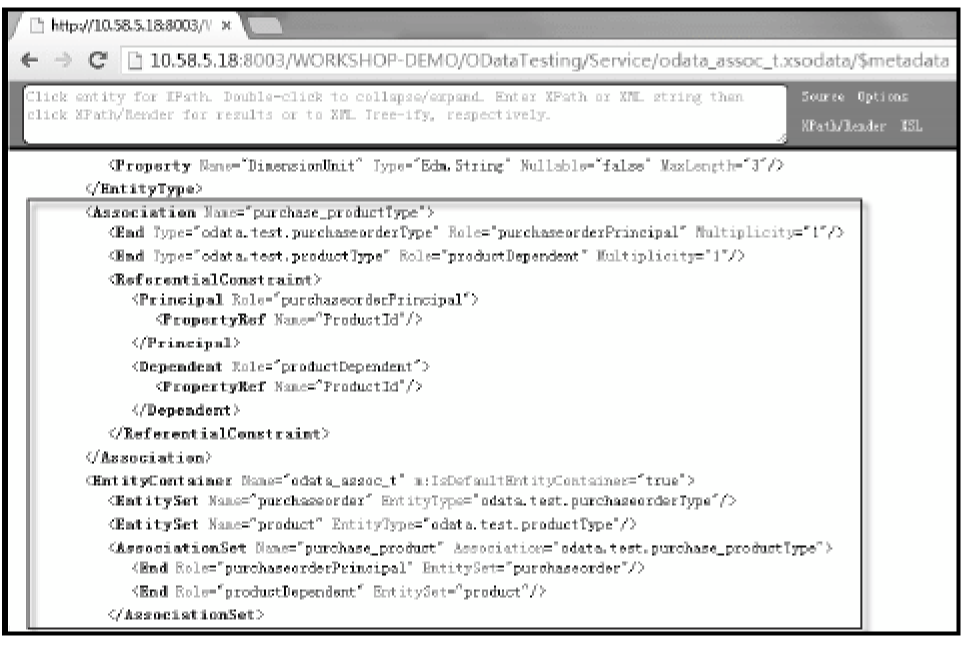


图 11-89

黑框内标出了我们设定的“purchase_product”关联的内容在浏览器中的显示结果。通过这个关联实体，我们就能实现通过“purchaseorder”数据源的某条数据来关联“product”数据源与之对应的某条数据的功能。

(4) 对“purchaseorder”数据源进行前两条数据的数据请求。由于“purchaseorder”数据源包含了上万条数据，我们只取前两条作为范例。这里就要用到前面介绍的“$top”参数：“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/odata_assoc_t.xsodata/purchaseorder/?$top=2”。为节省页面，我们只列出第一条记录的返回值部分(见图 11-90)。

```
<link rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/npur_product"
type="application/atom+xml;type=entry" title="npur_product"
href="purchaseorder(PurchaseOrderId='0300000000',PurchaseOrderItem='0000000010')/npur_product"/>
<category term="odata.test.purchaseorderType"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<content type="application/xml">
  <m:properties>
    <d:PurchaseOrderId m:type="Edm.String">0300000000</d:PurchaseOrderId>
    <d:PurchaseOrderItem m:type="Edm.String">0000000010</d:PurchaseOrderItem>
    <d:ProductId m:type="Edm.String">HT-1000</d:ProductId>
    <d:NoteId m:type="Edm.String" m:null="true"/>
    <d:Currency m:type="Edm.String">EUR</d:Currency>
    <d:GrossAmount m:type="Edm.Decimal">1137.64</d:GrossAmount>
    <d:NetAmount m:type="Edm.Decimal">956</d:NetAmount>
    <d:TaxAmount m:type="Edm.Decimal">181.64</d:TaxAmount>
    <d:Quantity m:type="Edm.Decimal">1</d:Quantity>
    <d:QuantityUnit m:type="Edm.String">EA</d:QuantityUnit>
    <d:DeliveryDate m:type="Edm.DateTime">2012-12-04T00:00:00.0000000</d:DeliveryDate>
  </m:properties>
</content>
```

图 11-90

请大家注意上图红框标出的部分，此部分是我们在 OData 服务中定义的导航属性，其中最后的超链接“href="purchaseorder(PurchaseOrderId='0300000000',PurchaseOrderItem='0000000020')/npur_product"”则提供了导航到“product”数据源中“ProductId”=“HT-100”的条目中去的功能。我们把该超链接添加到 OData 服务中，并将得到的 URL 输入到浏览器中查看：“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/odata_assoc_t.xsodata/purchaseorder(PurchaseOrderId='0300000000',PurchaseOrderItem='0000000020')/npur_product/”(见图 11-91)。

```
<id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_assoc_t.xsodata/product('HT-1000')</id>
<title type="text"/>
<author>
  <name/>
</author>
<category term="odata.test.productType"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<content type="application/xml">
  <m:properties>
    <d:ProductId m:type="Edm.String">HT-1000</d:ProductId>
    <d:TypeCode m:type="Edm.String">PR</d:TypeCode>
    <d:Category m:type="Edm.String">Notebooks</d:Category>
    <d:CreatedBy m:type="Edm.String">0000000033</d:CreatedBy>
    <d:CreatedAt m:type="Edm.DateTime">2012-10-03T00:00:00.0000000</d:CreatedAt>
    <d:ChangedBy m:type="Edm.String">0000000033</d:ChangedBy>
```

图 11-91

从截图可以看出此 URL 将“product”数据源中“ProductId”=“HT-100”的条目暴露了出来。

三、为 OData 服务定义关键字

前面我们介绍的很多 OData 服务都是将常见的数据表作为需暴露的数据源的，由于数据表在创建时就会指定关键字，因此 OData 服务在暴露这类数据源时会自动引用其预定义的关键字。例如我们前面引用的“purchaseorder”数据源，其关键字“PurchaseOrderId”和“PurchaseOrderItem”是继承于所重命名的数据表“sap.hana.democontent.epm.data::purchaseOrderItem”的，如图 11-92 所示。

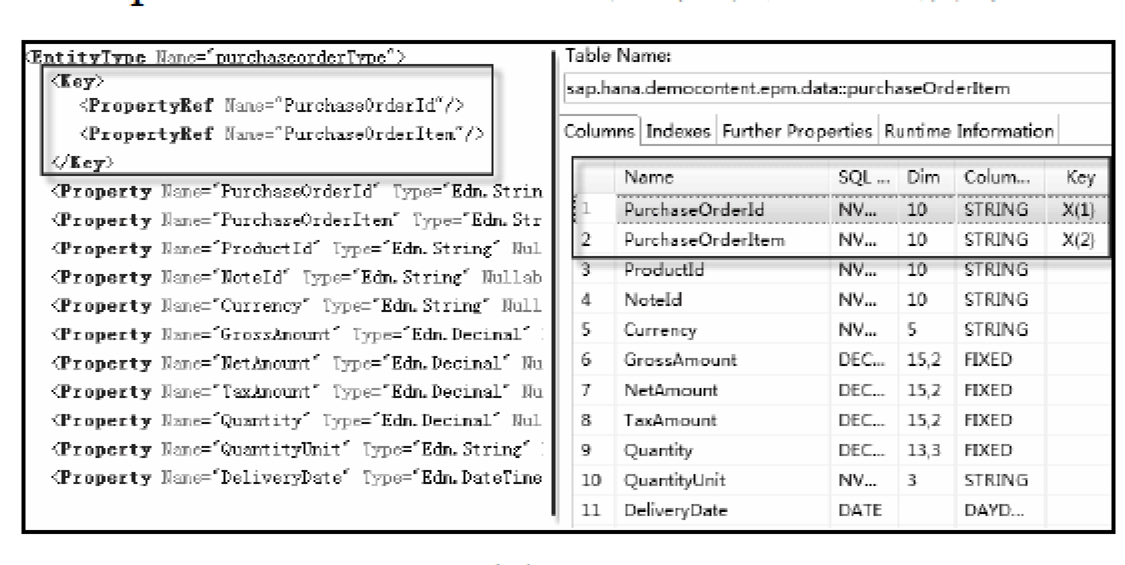


图 11-92

不同于这些数据表的是，如果我们在 OData 服务中指定了数据源是 SAP HANA Studio 的信息视图，如属性视图、分析视图或者计算视图，我们就需要考虑为这类的数据源添加关键字来对这些实体进行标识。

在 OData 服务中，你可以通过两种方法为这类的数据源定义关键字，一种方法是将数据源中已经存在的列定义为关键字；另一种方法是为这个数据源自动生成一个列作为关键字。接下来我们通过实例分别介绍这两种方法。

(1) 在项目中新建一个 OData 服务定义文件“odata_key_demo.xsodata”(见图 11-93)。

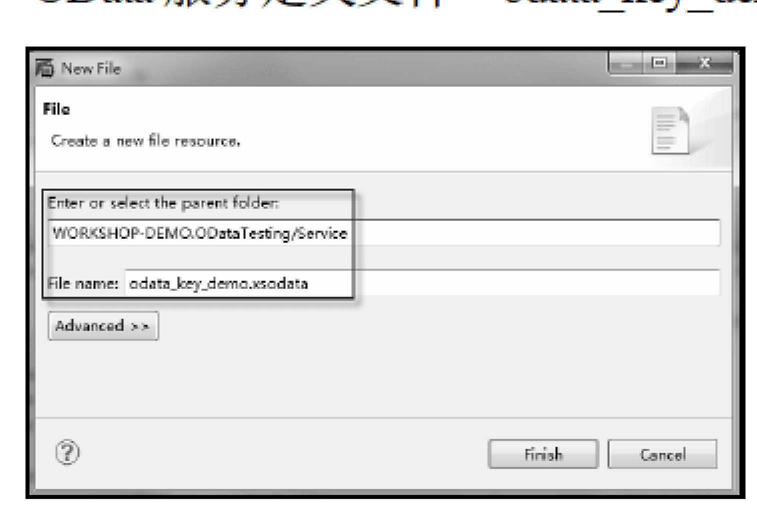


图 11-93

(2) 在编辑区输入如图 11-94 所示的代码。

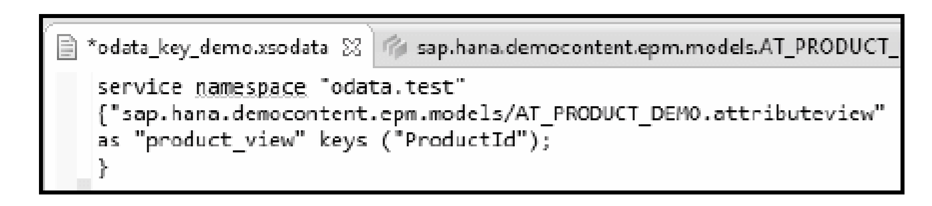
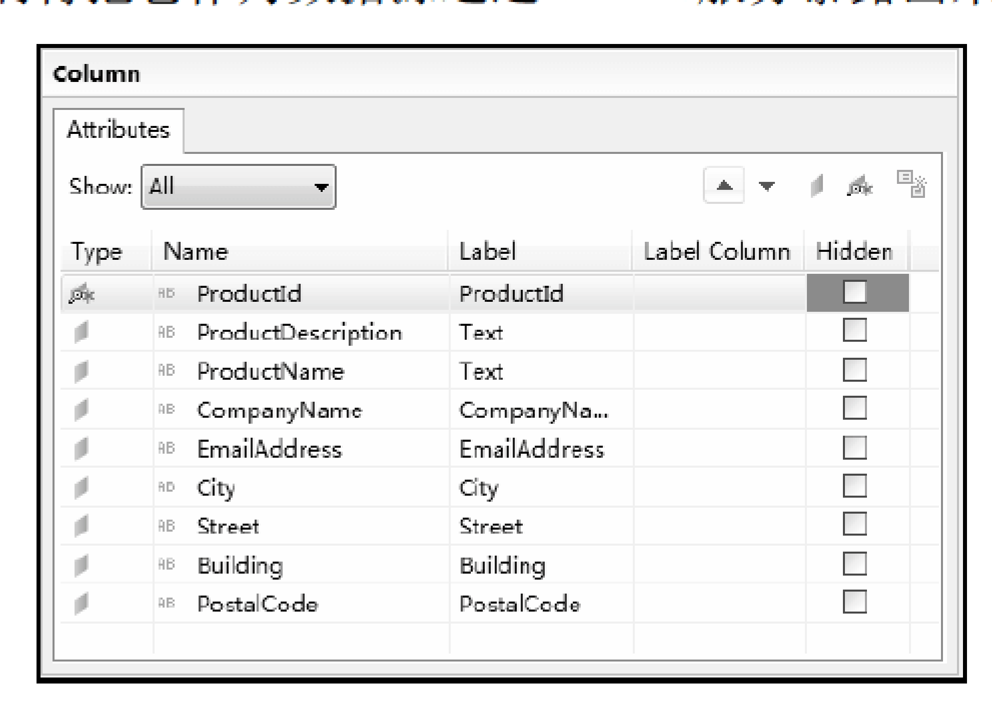

```
service namespace "odata.test"
{"sap.hana.democontent.epm.models/AT_PRODUCT_DEMO.attributeview"
as "product_view" keys ("ProductId");
}
```

图 11-94

假如你还有印象，我们在这段代码中引用的属性视图即为我们在介绍 SAP HANA Studio 建模时创建的属性视图的范例“AT_PRODUCT_DEMO”，当时我们创建好的视图模型如图 11-95 所示，这个属性视图包含了产品(Product)的全部属性信息，而在这里我们将把它作为数据源通过 OData 服务暴露出来。



Type	Name	Label	Label Column	Hidden
	ProductId	ProductId		
	ProductDescription	Text		
	ProductName	Text		
	CompanyName	CompanyNa...		
	EmailAddress	EmailAddress		
	City	City		
	Street	Street		
	Building	Building		
	PostalCode	PostalCode		

图 11-95

接下来解释一下刚刚所输入的代码含义，首先我们声明要引用“sap.hana.democontent.epm.models”路径下的“AT_PRODUCT_DEMO.attributeview”，并将它重命名为“product_view”。随后我们为此数据源定义“ProductId”为关键字。在 OData 服务定义中，如果你要定义多个关键字，则需要将多个关键字用逗号分隔开，例如我们定义“ProductId”、“City”和“PostalCode”为关键字的代码(见图 11-96)。

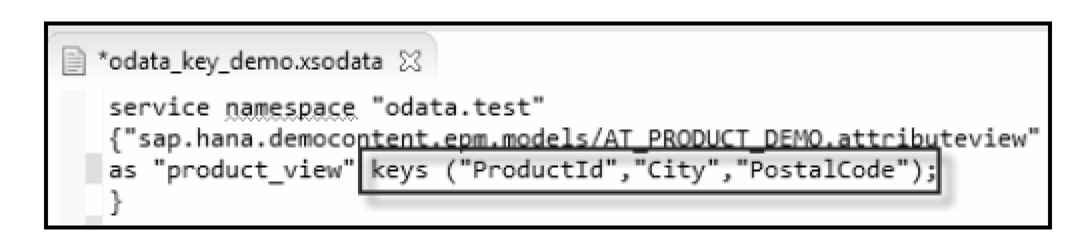

```
service namespace "odata.test"
{"sap.hana.democontent.epm.models/AT_PRODUCT_DEMO.attributeview"
as "product_view" keys ("ProductId","City","PostalCode");
}
```

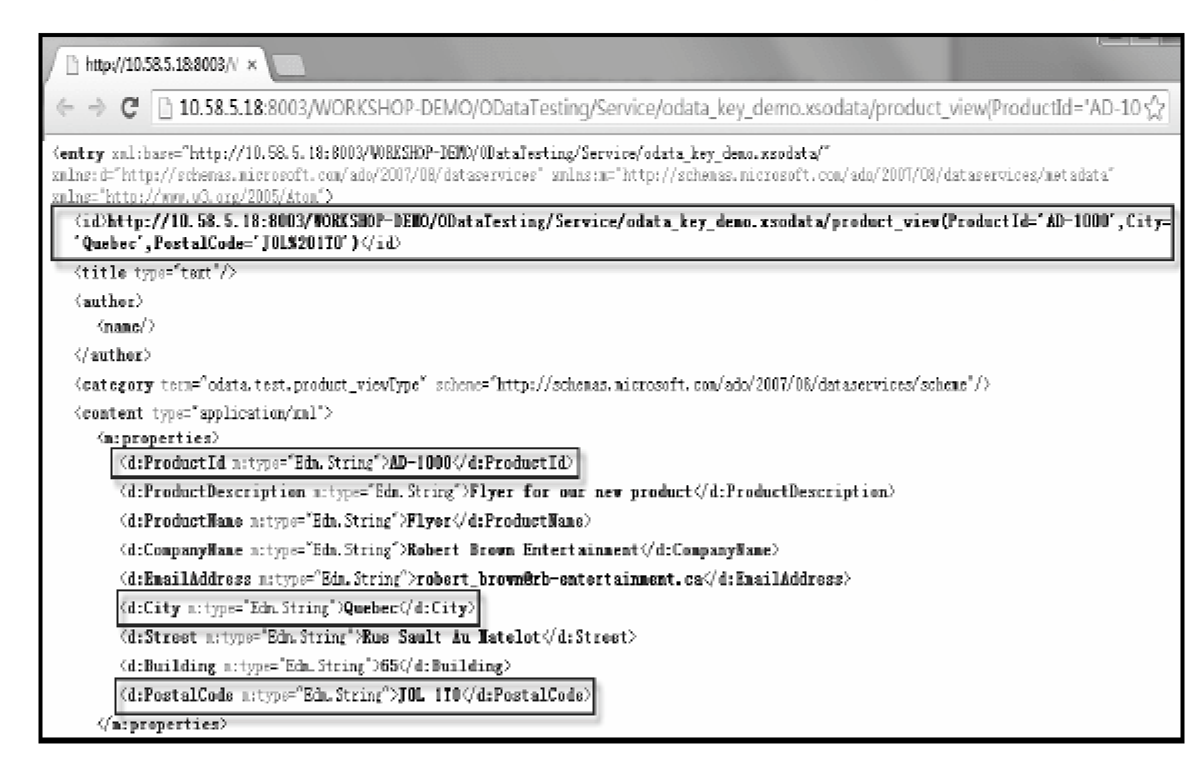
图 11-96

(3) 提交并激活“odata_key_demo.xsodata”文件，在浏览器中查看其元数据：“http://xxxxx/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/$metadata”(见图 11-97)。

```
<EntityType Name="product_viewType">
  <Key>
    <PropertyRef Name="ProductId"/>
    <PropertyRef Name="City"/>
    <PropertyRef Name="PostalCode"/>
  </Key>
  <Property Name="ProductId" Type="Edm.String" Nullable="true" MaxLength="10"/>
  <Property Name="ProductDescription" Type="Edm.String" Nullable="true" MaxLength="1024"/>
  <Property Name="ProductName" Type="Edm.String" Nullable="true" MaxLength="1024"/>
  <Property Name="CompanyName" Type="Edm.String" Nullable="true" MaxLength="80"/>
  <Property Name="EmailAddress" Type="Edm.String" Nullable="true" MaxLength="255"/>
  <Property Name="City" Type="Edm.String" Nullable="true" MaxLength="40"/>
  <Property Name="Street" Type="Edm.String" Nullable="true" MaxLength="60"/>
  <Property Name="Building" Type="Edm.String" Nullable="true" MaxLength="10"/>
  <Property Name="PostalCode" Type="Edm.String" Nullable="true" MaxLength="10"/>
</EntityType>
```

图 11-97

从截图可以看出我们定义的关键字已经被正确地反映到 OData 服务的元数据中去了。这样我们就可以通过我们自定义的关键字来对数据源的单个实体进行查询显示，例如我们需要找出某实体的关键字满足“ProductId='AD-1000',City='Quebec', PostalCode='J0L 1T0'”，则可以通过 OData 服务与超链接“/product_view(ProductId='AD-1000',City='Quebec',PostalCode='J0L 1T0'”来实现(见图 11-98)。



```
http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view(ProductId='AD-10
<entry xml:base="http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns="http://www.w3.org/2005/Atom">
  <id>http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view(ProductId='AD-1000',City=
  'Quebec',PostalCode='J0L%201T0')</id>
  <title type="text"/>
  <author>
    <name/>
  </author>
  <category term="odata.test.product_viewType" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
  <content type="application/xml">
    <m:properties>
      <d:ProductId m:type="Edm.String">AD-1000</d:ProductId>
      <d:ProductDescription m:type="Edm.String">Flyer for our new product</d:ProductDescription>
      <d:ProductName m:type="Edm.String">Flyer</d:ProductName>
      <d:CompanyName m:type="Edm.String">Robert Brown Entertainment</d:CompanyName>
      <d:EmailAddress m:type="Edm.String">robert_brown@rb-entertainment.ca</d:EmailAddress>
      <d:City m:type="Edm.String">Quebec</d:City>
      <d:Street m:type="Edm.String">Rue Sault Au Matelot</d:Street>
      <d:Building m:type="Edm.String">65</d:Building>
      <d:PostalCode m:type="Edm.String">J0L 1T0</d:PostalCode>
    </m:properties>
```

图 11-98

需要注意的是，由于 OData 服务不会检查关键字的唯一性，因此大家在定义关键字时一定要保证定义的关键字是能唯一能标识数据实体的，即同一关键字不可以对应两个或以上的数据实体。这与定义数据表的关键字概念是一样的。

(4) 自动生成关键字。按照图 11-99 代码修改“odata_key_demo.xsodata”文件。

```
odata_key_demo.xsodata
service namespace "odata.test"
{"sap.hana.democontent.epm.models/AT_PRODUCT_DEMO.attributeview"
as "product_view" keys generate local "KeyID";
}
```

图 11-99

有时候某些数据对象，尤其是计算视图和做了聚合的数据表，我们很难对它们



基于已经存在的列定义关键字，这个时候我们可以使用 OData 服务自动生成一个本地的关键字(Local Key)，这个关键字从数字“1”开始依次递增，直到数据源数据的末尾结束。由于该关键字只在调用 OData 服务时临时生成，当我们将 OData 服务调用进程结束后这个临时关键字会自动消失，正是这种特性，导致我们不可以通过这个临时的关键字来像上面的例子中那样定位到数据实体，因为每次调用 OData 服务时其暴露的数据源所包含的数据实体的临时关键字都会重新生成，这造成每次结果都不一样。

我们在本步输入的代码和在第二步输入的代码唯一的不同是我们定义了一个本地的名称为“KeyID”的关键字，即当我们需要定义本地关键字时，需要使用“keys generate local+关键字名称”的语法结构。

(5) 将修改过的“odata_key_demo.xsodata”文件提交并激活，在浏览器中查看其元数据(见图 11-100)。

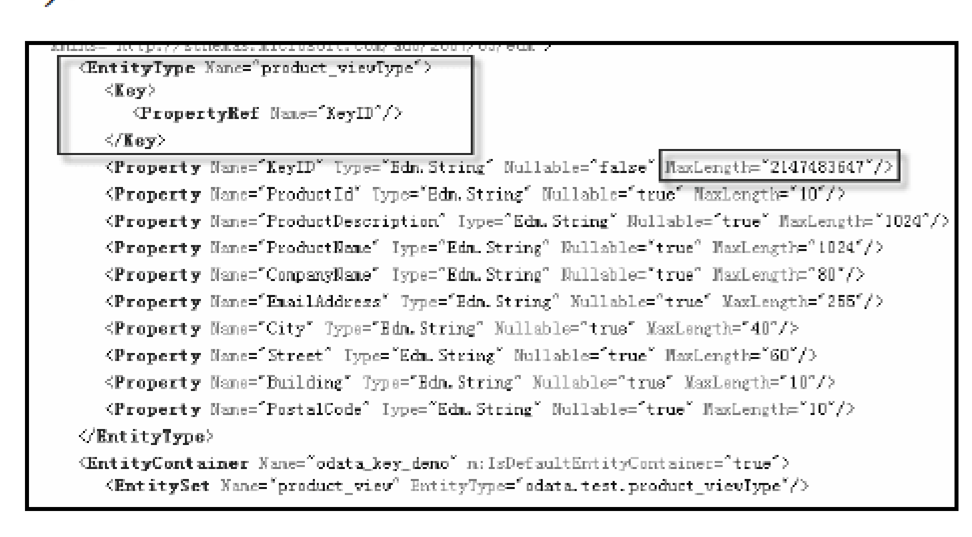


图 11-100

细心的读者如果返回到第 3 步比较一下这两张截图的不同，应该可以看出这个 OData 服务所暴露的数据源的关键字已经改为“KeyID”，并且其最大长度为“2147483647”，即只要数据源的数据条目数不超过 2 147 483 647 条，这个本地关键字都是可用的。

下面我们通过调用“product_view”来看看这个本地关键字在数据暴露时的显示(见图 11-101)。

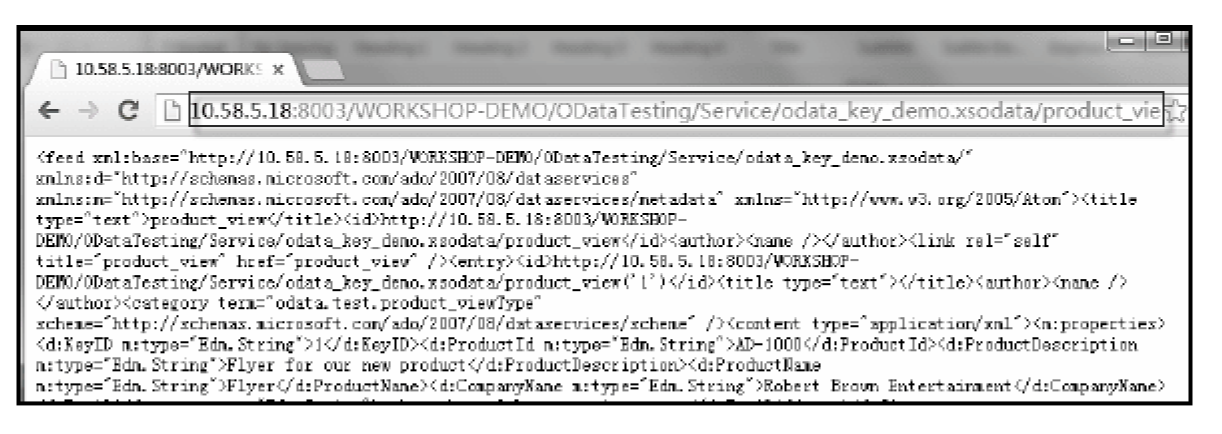


图 11-101



在默认的“Atom”格式下，浏览器无法对返回的数据进行解析，我们切换到“JSON”格式看看结果(见图 11-102)。

10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view?$format=json

```
{
 - d: {
   - results: [
     - {
       - __metadata: {
           uri: "http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view('1')",
           type: "odata.test.product_viewType"
         },
         KeyID: "1",
         ProductId: "AD-1000",
         ProductDescription: "Flyer for our new product",
         ProductName: "Flyer",
         CompanyName: "Robert Brown Entertainment",
         EmailAddress: "robert_brown@rb-entertainment.ca",
         City: "Quebec",
         Street: "Rue Sault Au Matelot",
         Building: "65",
         PostalCode: "J0L 1T0"
       },
     - {
       - __metadata: {
           uri: "http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view('2')",
           type: "odata.test.product_viewType"
         },
         KeyID: "2",
         ProductId: "HT-1000",
         ProductDescription: "Notebook Basic 15 with 1.7GHz - 15 XGA - 1024MB DDR2 SDRAM - 40GB Hard Disc",
         ProductName: "Notebook Basic 15",
         CompanyName: "SAP",
         EmailAddress: "karl.mueller@sap.com",
         City: "Walldorf",
         Street: "Dietmar-Hopp-Allee",
         Building: "16",
         PostalCode: "69190"
       },
```


图 11-102

在“JSON”格式下，我们可以看到每条记录都自动增加了名为“KeyID”的列，并且该列的数值从“1”开始自动递增。那么如果我们像以前那样点击实体关键字标识 URL(如图 11-103 所示)会有什么结果呢？


```
{
 - d: {
   - results: [
     - {
       - __metadata: {
           uri: "http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/product_view('1')"
           type: "odata.test.product_viewType"
         },
         KeyID: "1",
```


图 11-103



在我们以前的演示中，这个 URL 会将关键字所标识的数据实体内容暴露出来，而从图 11-104 中可以看出，本地关键字标识的实体是无法通过这个 URL 暴露的，当我们点击这个 URL 会得到“找不到资源”的错误信息。



图 11-104

这个结果正好印证了我们刚才所讲的本地关键字不可以用来定位数据实体的说

法，因此如果你需要使用关键字来在 OData 服务中暴露单独的数据实体，请慎用本地关键字。

四、为 OData 服务定义聚合

在介绍 OData 服务聚合(Aggregation)定义之前，我们先来做个小测试。首先我们把上例中暴露的数据源由属性视图变成分析视图，看看 OData 服务是否还能正常运行。

修改后的代码如图 11-105 所示。

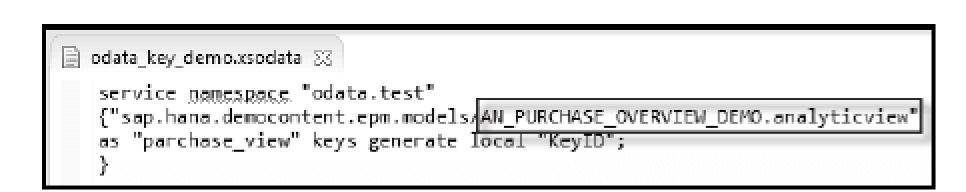


图 11-105

我们调用在以前章节中做好的“AN_PURCHASE_OVERVIEW_DEMO”分析视图作为需暴露的数据源，而作为关键字我们依然沿用自动生成的本地关键字。提交并激活此 OData 服务文件后。我们在浏览器中打开它的元数据 URL(见图 11-106)。

```
<EntityType Name="parchase_viewType">
  <Key>
    <PropertyRef Name="KeyID"/>
  </Key>
  <Property Name="KeyID" Type="Edm.String" Nullable="false" MaxLength="2147483647"/>
  <Property Name="PurchaseOrderId" Type="Edm.String" Nullable="true" MaxLength="10"/>
  <Property Name="Currency" Type="Edm.String" Nullable="true" MaxLength="5"/>
  <Property Name="QuantityUnit" Type="Edm.String" Nullable="true" MaxLength="3"/>
  <Property Name="DeliveryDate" Type="Edm.DateTime" Nullable="true"/>
  <Property Name="ProductId" Type="Edm.String" Nullable="true" MaxLength="10"/>
  <Property Name="ProductDescription" Type="Edm.String" Nullable="true" MaxLength="1024"/>
  <Property Name="ProductName" Type="Edm.String" Nullable="true" MaxLength="1024"/>
  <Property Name="CompanyName" Type="Edm.String" Nullable="true" MaxLength="80"/>
  <Property Name="EmailAddress" Type="Edm.String" Nullable="true" MaxLength="255"/>
  <Property Name="City" Type="Edm.String" Nullable="true" MaxLength="40"/>
  <Property Name="Street" Type="Edm.String" Nullable="true" MaxLength="60"/>
```

图 11-106

由图 11-106 我们可以看出 OData 元数据是正常的，那么我们再尝试调用到数据实体 URL(见图 11-107)。

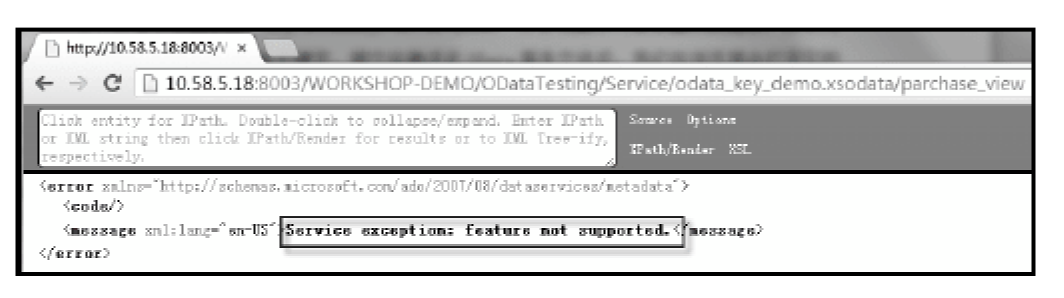
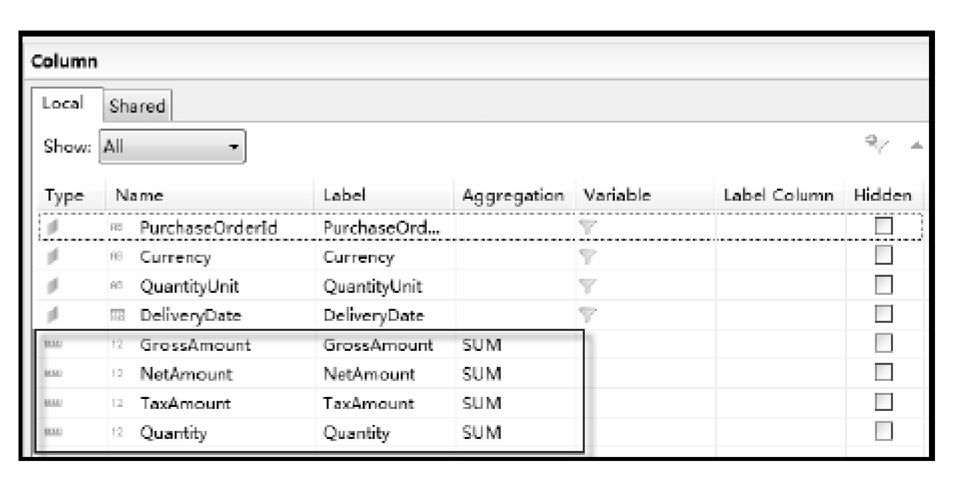


图 11-107

到这步我们发现数据实体不能被显示，OData 提交了服务异常(Service exception)的错误。那导致服务异常的原因是什么呢？我们知道，在定义分析视图或者计算视图时，我们通常会在建立视图时引用的度量数据相关的数据列上面定义聚合，




如图 11-108 所示。



Type	Name	Label	Aggregation	Variable	Label Column	Hidden
	PurchaseOrderId	PurchaseOrd...				
	Currency	Currency				
	QuantityUnit	QuantityUnit				
	DeliveryDate	DeliveryDate				
	GrossAmount	GrossAmount	SUM			
	NetAmount	NetAmount	SUM			
	TaxAmount	TaxAmount	SUM			
	Quantity	Quantity	SUM			

图 11-108

因此，当我们需要暴露分析视图或者计算视图时，我们就需要在 OData 服务定义文件中为这类的数据源定义聚合，否则我们在调用数据实体时，OData 服务就会提交服务异常的错误。我们在这里将这点着重指出是因为这种错误在开发中非常常见却难以被发现。因为从这个例子可以看出，我们在建立(分析视图/计算视图)OData 服务时，即便我们不在定义文件中定义聚合参数，其定义文件也能顺利激活并被调用的，并且其元数据也被正常显示，只有当你想要调用其数据实体时，才会发现服务异常的错误。

为分析视图或者是计算视图作为数据源的 OData 服务定义聚合非常简单，只需要在定义文件代码的末尾加上“aggregates always”关键字，如图 11-109 所示。

```
odata_key_demo.xsodata
service namespace "odata.test"
{"sap.hana.democontent.epm.models/AN_PURCHASE_OVERVIEW_DEMO.analyticview"
as "parchase_view" keys generate local "KeyID" aggregates always;
}
```

图 11-109



将新修改的 OData 定义文件提交并激活，我们再次在浏览器中调用其数据实体(见图 11-110)。

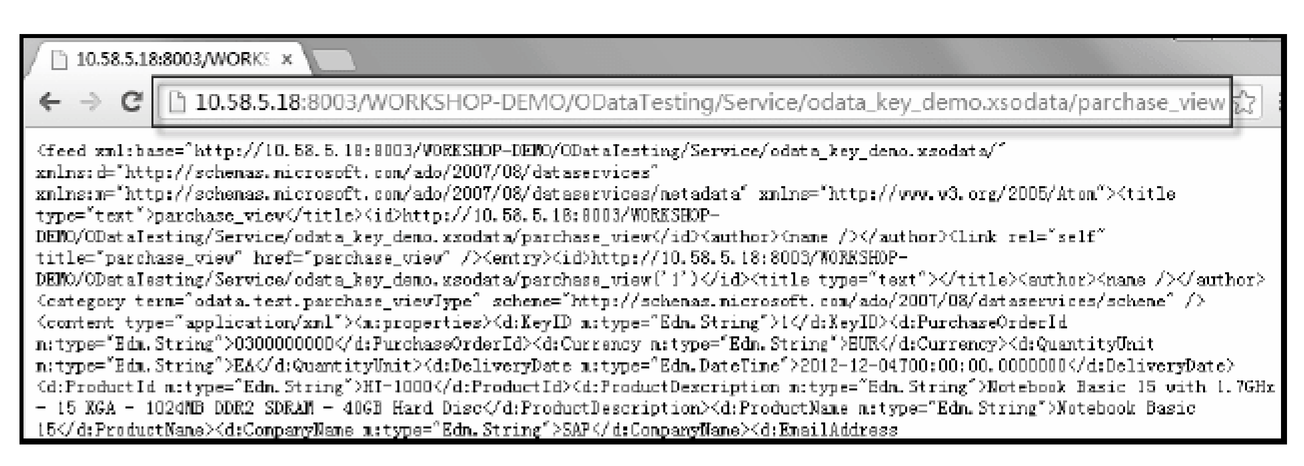


图 11-110

在定义文件代码加入聚合关键字后，我们就能顺利地调出此分析视图包含的所

有数据实体了，为方便观察我们使用“$format=JSON”参数将数据格式转换为“JSON”格式(见图 11-111)。


```
{
 - d: {
   - results: [
     - {
       - __metadata: {
           uri: "http://10.58.5.18:8003/WORKSHOP-DEMO/ODataTesting/Service/odata_key_demo.xsodata/purchase_view('1')",
           type: "odata.test.purchase_viewType"
         },
         KeyID: "1",
         PurchaseOrderId: "0300000000",
         Currency: "EUR",
         QuantityUnit: "EA",
         DeliveryDate: "/Date(1354579200000)/",
         ProductId: "HT-1000",
         ProductDescription: "Notebook Basic 15 with 1.7GHz - 15 XGA - 1024MB DDR2 SDRAM - 40GB Hard Disc",
         ProductName: "Notebook Basic 15",
         CompanyName: "SAP",
         EmailAddress: "karl.mueller@sap.com",
         City: "Walldorf",
         Street: "Dietmar-Hopp-Allee",
         Building: "15",
         PostalCode: "69190",
         GrossAmount: "9101.12",
         NetAmount: "7648",
         TaxAmount: "1453.12",
         Quantity: "8"
       },
```


图 11-111

可能会有读者问，聚合定义只能在分析视图或者计算视图作为数据源时使用吗？如果我需要在常用的数据表中也定义聚合怎么办？那么接下来的内容会很好地解答这两个问题。

首先 OData 服务的聚合关键字同样也适用于包含有度量数据的数据表，其次当我们需要对此类数据表在 OData 服务中定义聚合时，就需要明确指出哪列数据列是需要在数据暴露时做聚合操作的。下面我们用一个具体的例子来说明这点。

在这个例子中我们要用到的数据表是“sap.hana.democontent.epm.data::purchaseOrder”，这个数据表在以前我们创建分析视图时引用过，它包含了诸如“总额”、“净额”以及“税额”等度量数据。

为了更好地方便大家的理解，我们将该数据表的结构再次贴出来(见图 11-112)。

Table Name:
sap.hana.democontent.epm.data::purchaseOrder

Columns | Indexes | Further Properties | Runtime Information

	Name	SQL Data Type	Dim	Column Store Data Ty...	Key	Not Null
1	PurchaseOrderId	NVARCHAR	10	STRING	X(1)	X
2	CreatedBy	NVARCHAR	10	STRING		X
3	CreatedAt	DATE		DAYDATE		X
4	ChangedBy	NVARCHAR	10	STRING		
5	ChangedAt	DATE		DAYDATE		
6	NoteId	NVARCHAR	10	STRING		
7	PartnerId	NVARCHAR	10	STRING		
8	Currency	NVARCHAR	5	STRING		X
9	GrossAmount	DECIMAL	15,2	FIXED		X
10	NetAmount	DECIMAL	15,2	FIXED		X
11	TaxAmount	DECIMAL	15,2	FIXED		X

图 11-112

在本例中，我们对“TaxAmount”列在 OData 服务中做汇总的聚合操作，具体代码如图 11-113。

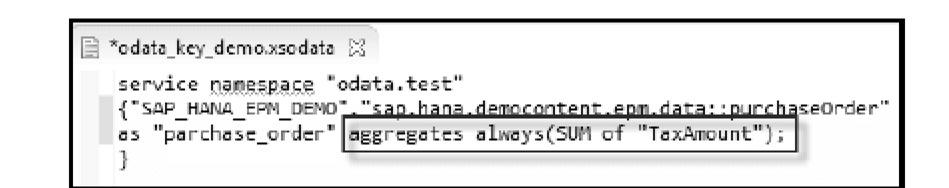


图 11-113

我们对“aggregates always”关键字增加了参数“SUM”(对“TaxAmount”字段)。“SUM”参数表明在 OData 服务中对需要聚合的列的操作方法，除了“SUM”外，我们还可以使用“MAX”、“MIN”以及“COUNT”等其他聚合方法。如果有多个列需要聚合，则需要使用逗号将各列的聚合定义代码分隔开，如图 11-114 所示。

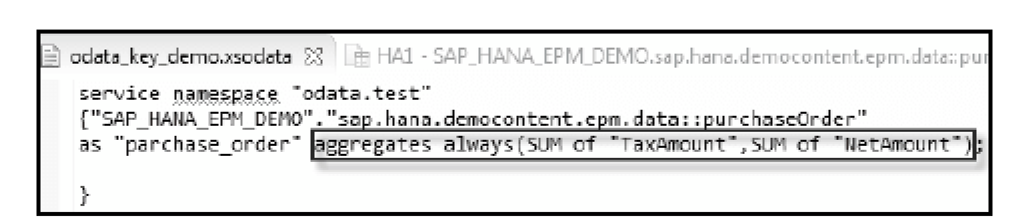


图 11-114

至此我们已经介绍完毕如何在 OData 服务中定义聚合操作，大家在今后的应用开发过程中可以自行多做练习，就能熟练掌握这一技巧了。

本章小结与练习

通过本章的学习，相信大家对于 SAP HANA XS 中的 OData 服务已经有了大概的认识。大家应该能够自己创建简单的 OData 服务并能够暴露一些复杂的数据源。

请大家务必注意，在创建 OData 服务之前，请确保你已经为 Odata 服务创建了相应的工作区和项目，并且 SAP HANA XS 服务已经被激活。而对于需要输出的数据来讲，请确保你已经为这些数据建立了 Schema 以及相应的数据表结构。

在你调用 Odata 服务出现错误时，请查看你新建的节点是否被正确赋予你调用 Odata 服务的用户，即此用户是否有查询、修改或者删除的权限。

练习

1. 请尝试激活你所使用的 SAP HANA 服务器的 OData 服务，并为今后的练习准备相应的工作区和其他基础环境。
2. 请基于本书的 EPM 实例，暴露其中的任一数据表。
3. 请尝试暴露你在前面章节所建立的属性视图。
4. 如果你创建了分析视图，请尝试使用聚合参数暴露其数据。

第十二章　编写 SAP HANA 服务器端应用

SAP HANA 扩展应用服务(SAP HANA XS)为应用程序和应用开发者提供了一种我们称之为服务器控件模式(Consumption Model，由 HTTP 协议暴露)来访问 SAP HANA 数据库的方法。这种服务器控件模式通常是指在 SAP HANA XS 服务器端由 JavaScript 编写的服务器端应用(Server-Side Application)。我们称这种在 SAP HANA XS 服务器端使用的 JavaScript 为 SAP HANA XSJS，而由这种 JavaScript 编写的应用为 SAP HANA XS 服务器端应用。对于 SAP HANA XS 服务器端应用，SAP 专门为其开发了很多强有力的 API 控件，例如专门用于访问服务器进程的控件，或者是专门用来访问数据库的控件等。SAP HANA XS 服务器端应用通过调用这些 API 控件，可以响应客户端的 HTTP 请求来暴露经过授权的服务器数据。不同于我们在前一章介绍的 OData 服务，SAP HANA XS 服务器端应用不仅能够像 OData 服务一样通过 HTTP 协议来暴露数据源的数据，而且还能够对数据源的数据进行添加、修改以及删除的操作。正因为如此，这类应用具有能和 SAP HANA XS 服务器做交互及直接访问 SAP HANA 数据库的特性。在这一章里面我们将为大家详细介绍如何利用这种服务器端应用来通过网页浏览器或者其他 HTTP 终端暴露 SAP HANA 数据的方法。

第一节　创建简单的 XSJS 应用

毫无疑问，在所有编程语言相关的书籍中，“Hello World”总是作为第一个出现的范例，因为它简单易懂，又包含了几乎全部创建应用的必需步骤。在本小节中，我们也以编写最经典的“Hello World”应用开始我们的 SAP HANA XS 服务器端应用之旅。在接下来的示范中，我们会严格按照在 SAP HANA XS 服务器端创建应用的要求逐步向大家介绍每一步的流程，请大家在今后的开发中也参考下面的步骤来创建自己的应用。

一、创建交付单元

交付单元(Delivery-Unit)我们在前几章已经介绍过，它是将我们所有开发项目打包传输到其他系统的关键。因此我们不管在 SAP HANA Stuido 中做任何项目的开发，首先第一步就是创建这个项目的交付单元。请遵循以下步骤在 SAP HANA Studio 中创建交付单元。

1. 定义 Vendor ID

在创建交付单元之前，我们首先要在系统中维护你自己的 Vendor ID。Vendor ID 类似于交付单元的身份证，它是用来区分包裹在交付单元里的内容是来自哪个公司/团体。在这里我们推荐如果你有自己的域名的话，请使用该域名来作为你的 Vendor ID。例如你的域名为“MyCpmpanyXXX.com”，则这个域名就可以作为 Vendor ID 来更好地向客户标识交付单元来自哪里。

在 SAP HANA Studio 中，我们在“Administration Console”透视图中维护 Vendor ID。请登录 SAP HANA Studio，切换到“Administration Console”透视图，在“Administration”视图中选择标签页“Configuration”，在此标签页中定位到“indexserver.ini”→“repository”→“content_vendor”。双击“content_vendor”选项，在“Change Configuration Value”窗口的“New Value:”字段输入你自己的 Vendor ID。单击“Save”按钮保存输入。在本例中，我们使用“workshopdemo.com”来作为范例应用的交付单元 Vendor ID(见图 12-1)。

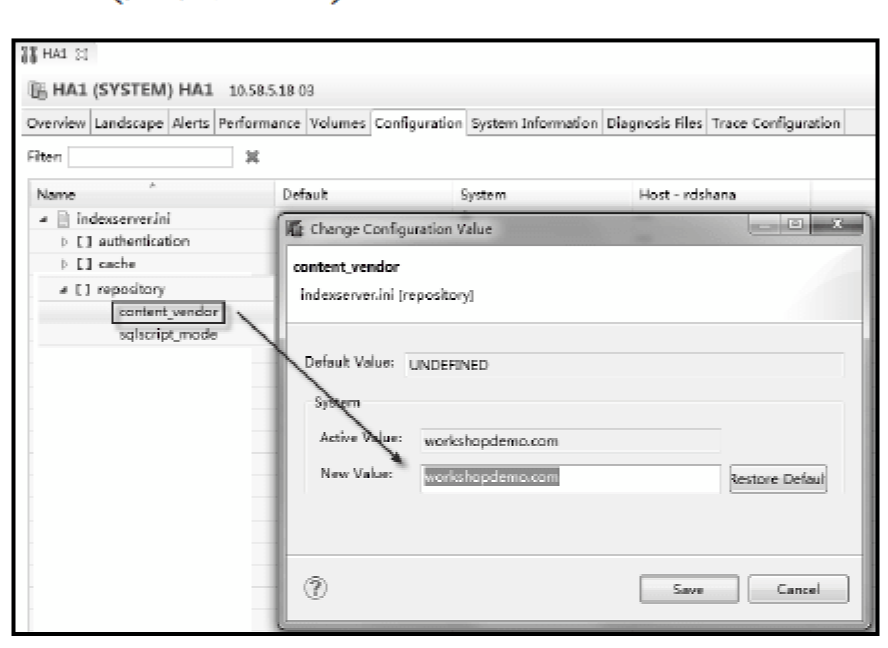


图 12-1

2. 创建交付单元

(1) 请登录 SAP HANA Studio，打开“Quick Launch”视图，在“Setup”区域

找到“Delivery Units”项，如图 12-2 所示。

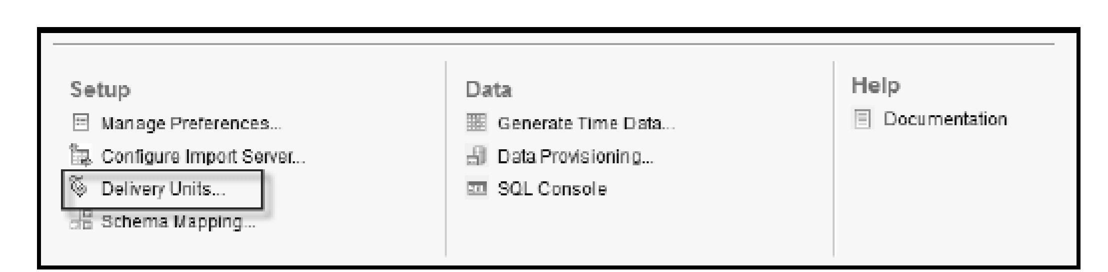


图 12-2

在图 12-3 所示的“Delivery Units”窗口中，你能找到目前在系统中存在的所有交付单元，这里我们会通过点击“Create...”按钮来新建一个交付单元。在以后的开发进程中，你还可以选择将新开发的项目分配给已经存在的交付单元，这种做法通常见于对已有项目的升级或者修改。

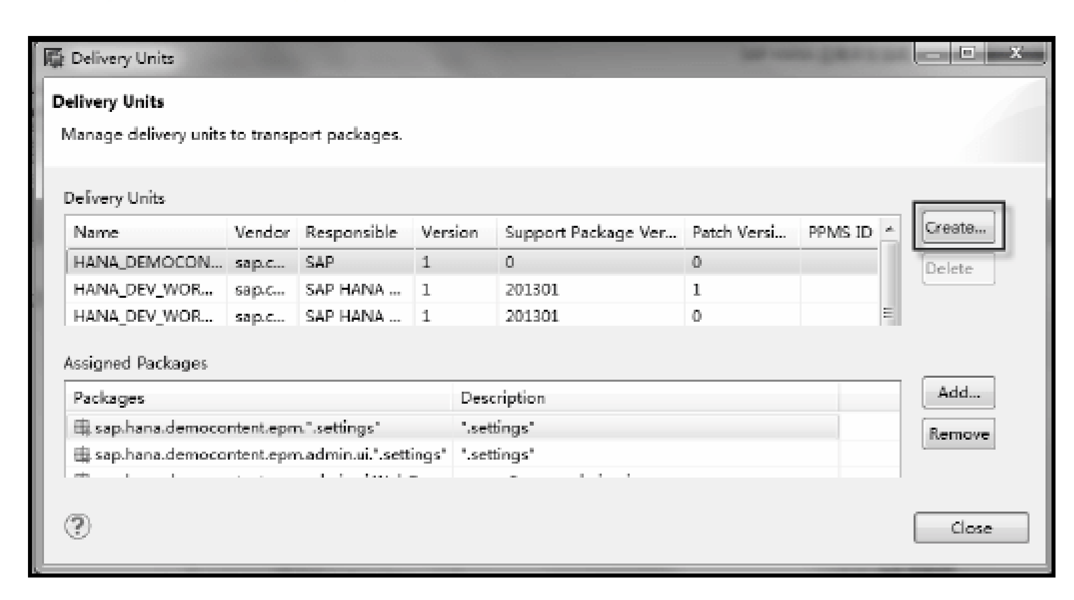


图 12-3

(2) 单击“Create...”按钮，在图 12-4 所示界面中填入所需内容。如果你对某一字段所填内容不太清楚，请单击下面方框标出的“?”按钮来得到系统的帮助。最后请单击“OK”按钮确认所有已输入信息。为了今后的操作，我们在这里会创建一个名为“WORKSHOP_APP_DEMO”的交付单元。

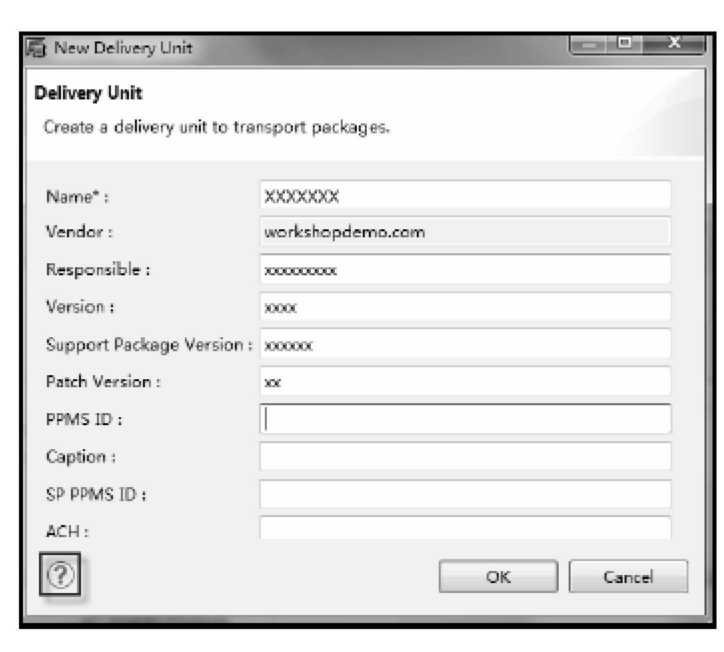


图 12-4

二、创建仓储工作区和项目

在第十一章我们已经介绍了如何创建仓储工作区(Repository Workspace)和项目(Project)，在这里需要强调的是，SAP HANA XS 服务器端应用与 OData 服务一样，都需要在 SAP HANA 服务器端存储并激活后才能使用，因此必须要为此类应用创建专属的仓储工作区和项目。

(1) 创建仓储工作区。在 SAP HANA Studio 菜单栏中定位到“File”→“New”→“New Repository Workspace”，在弹出的窗口中输入工作区的名称和根目录(见图 12-5)。

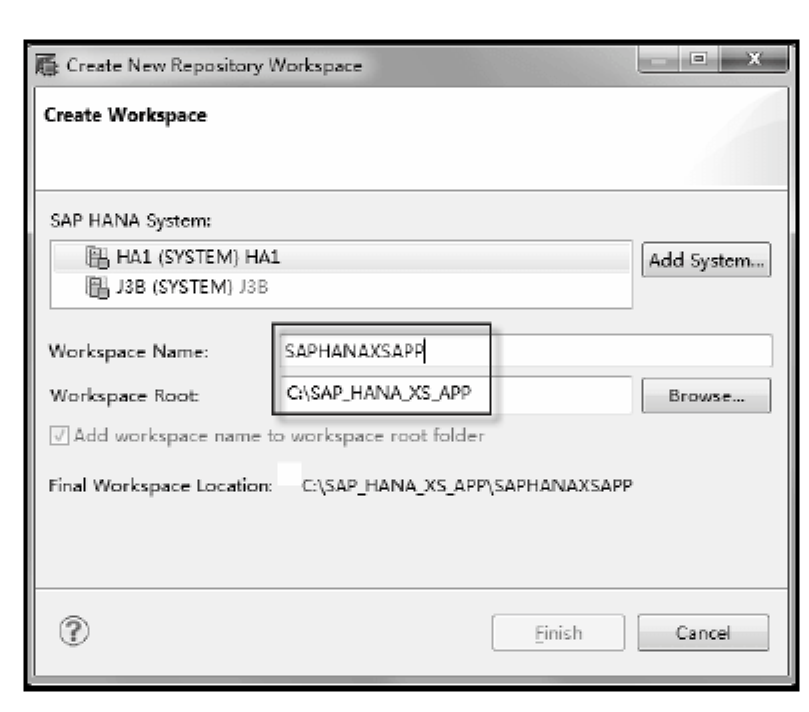


图 12-5

(2) 创建项目。在 SAP HANA Studio 菜单栏中定位到“File”→“New”→“Project...”，在弹出窗口中选中“XS Project”并单击“Next”按钮(见图 12-6)。

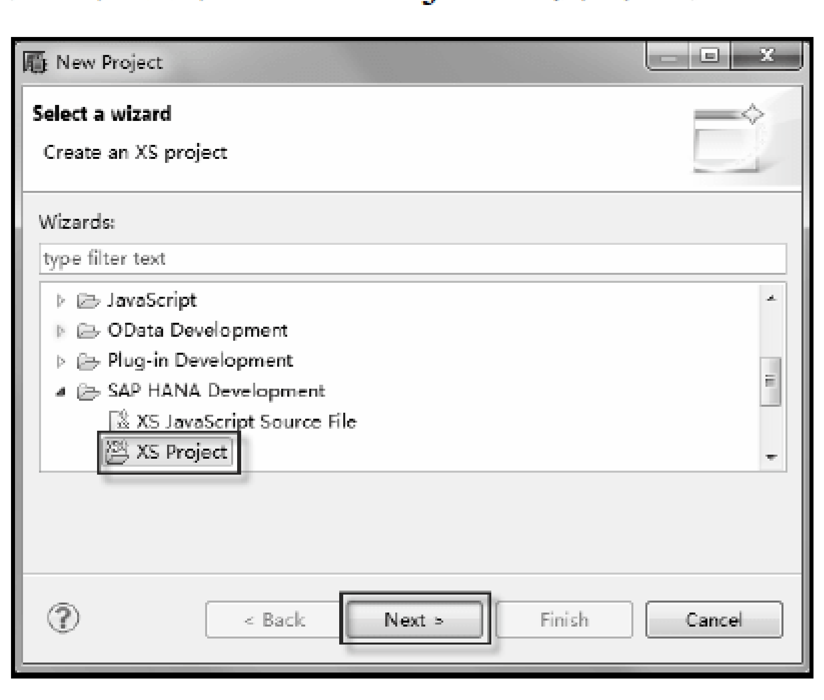


图 12-6

(3) 在下一窗口中输入项目名称“XSJSHELLOWORLD”，单击“Finish”按钮确



认(见图 12-7)。

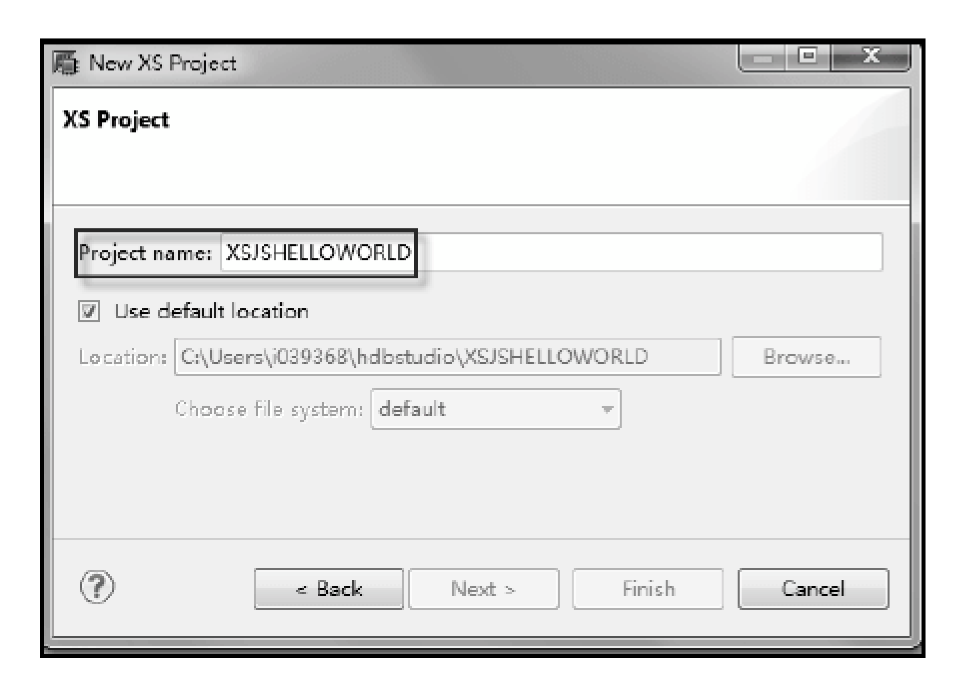


图 12-7

三、创建包

相信大家读到了这里，对包(Package)这个概念已经很熟悉了。在这一步我们要做的就是为 SAP HANA XS 服务器端应用创建包来存放其相关的所有对象(objects)。

(1) 在“SAP HANA Systems”视图中右击“Content”根目录，在弹出菜单中选择“New”→“Package...”选项(见图 12-8)。

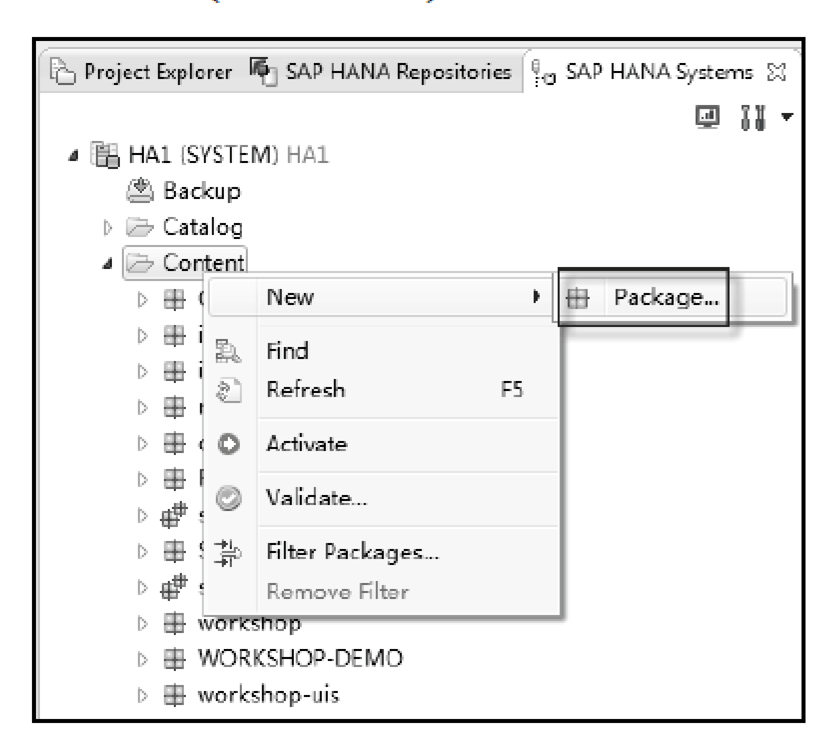


图 12-8

(2) 在弹出窗口中输入新包的名称和描述，选择刚才新建的交付单元“WORKSHOP_APP_DEMO”为此包裹的交付单元。在交付单元行我们能看见在前面维护的 Vendor ID 也同时被显示出来，单击“OK”按钮确认，在本步我们会创建一个名为“XSJSAPP”的包(见图 12-9)。

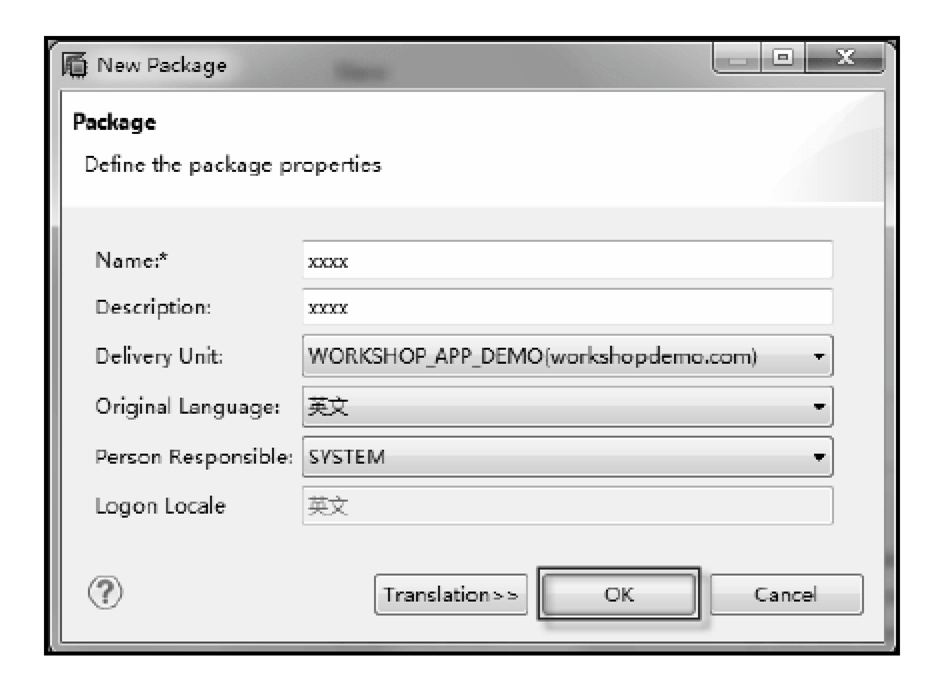


图 12-9

(3) 将新建的根包变为结构化包(对结构化/非结构化包的定义请参阅第八章)。在新建的“XSJSAPP”包上右击，在弹出菜单中选择“Edit”一项(见图 12-12)。

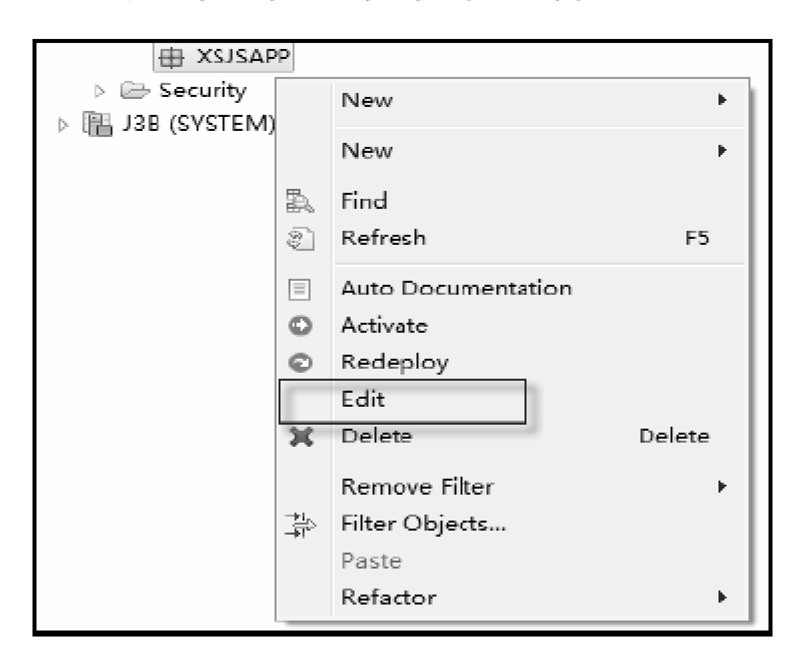


图 12-10

在弹出窗口中将“Structural”下拉选项变为“Yes”，单击“OK”按钮确认(见图 12-11)。

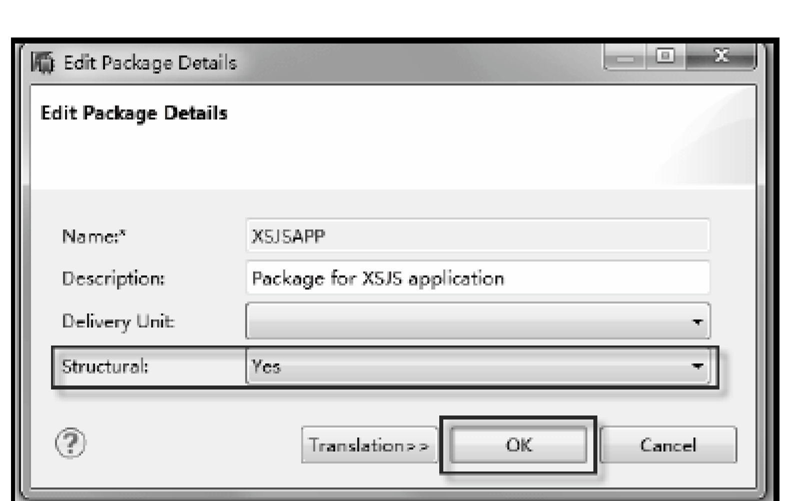


图 12-11

一般来讲我们新创建的包裹都为非结构化包，为了在包中更好地归类不同用途的

程序，我们需要把根包变成结构化包，然后新建子包来包含真正的应用。

(4) 在“XSJSAPP”根包处右击，选择“New”→“Package”，然后创建一个名为“MYAPP01”的子包，单击“OK”按钮确认(见图 12-12)。

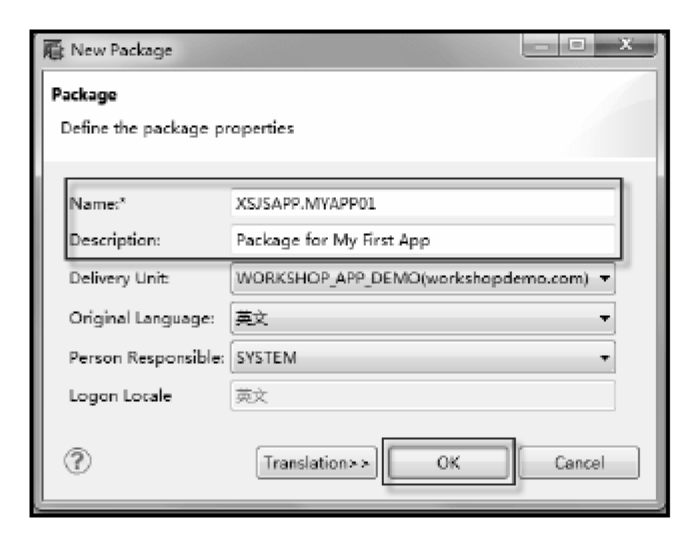


图 12-12

最终我们得到的包结构如图 12-13 所示。

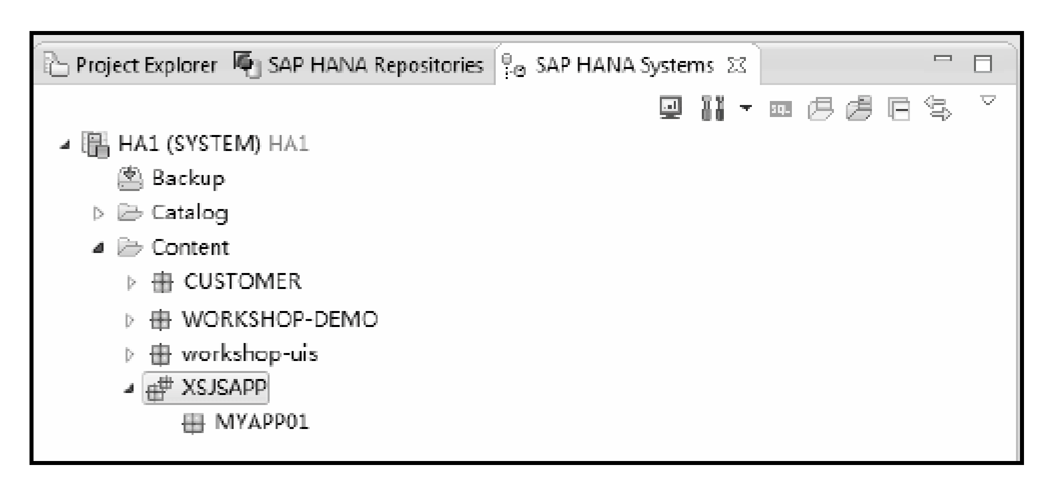


图 12-13

四、分享项目

创建好包后，我们就能分享我们新建的项目来同 SAP HANA 仓储(repository)作同步操作了。

(1) 在“Project Explorer”视图中右击“XSJSHELLOWORLD”项目，选择“Team”→“Share Project…”(见图 12-14)。

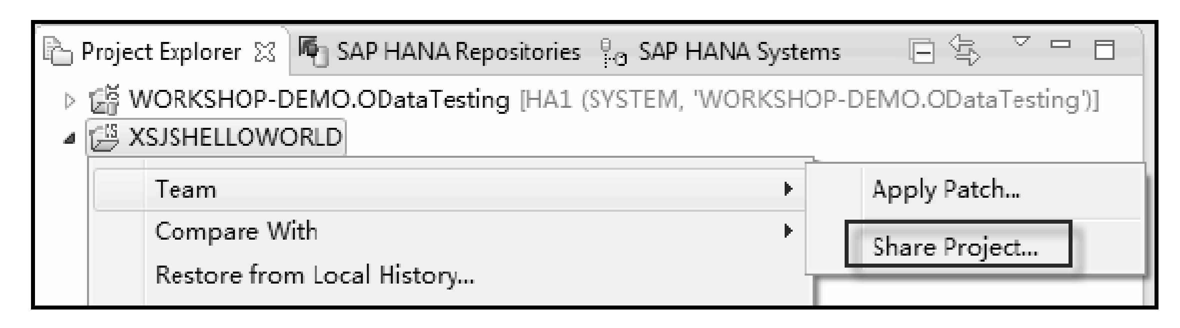


图 12-14

(2) 在弹出窗口中选择“SAP HANA Repository”并单击“Next”按钮(见图 12-15)。

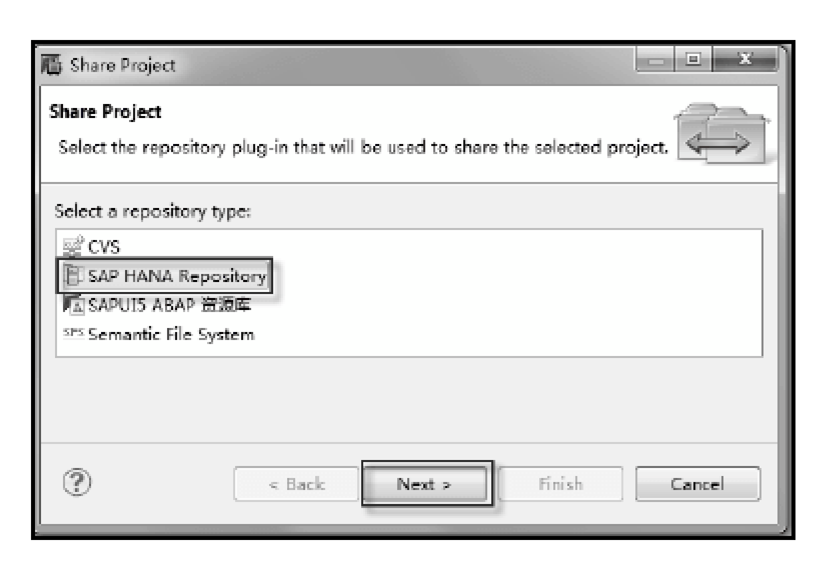


图 12-15

(3) 选择我们刚才新建的“SAP HANAXSAPP”工作区，并指定“MYAPP01”包为项目的包。单击“Finish”按钮确认(见图 12-16)。

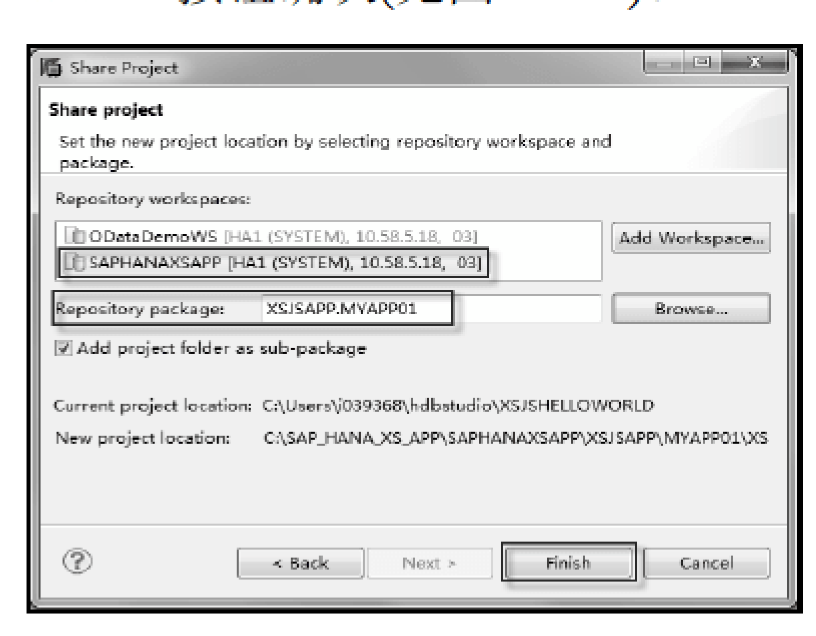


图 12-16

(4) 提交并激活分享后的项目。右击“XSJSHELLOWORLD”项目，选择“Team”→“Commit”/“Activate”(见图 12-17)。

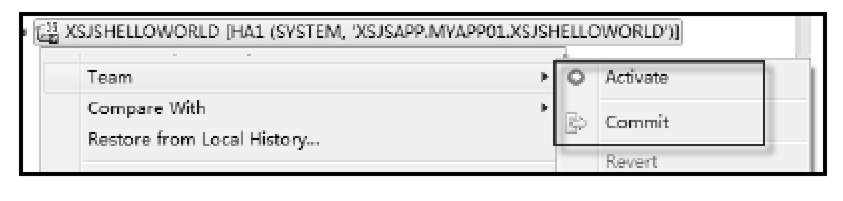


图 12-17

五、创建应用描述文件

当你在 SAP HANA XS 服务器端开发应用时，你必须要为你的应用创建应用描述文件(Application Descriptor File)。应用描述文件是用来指出应用在 SAP HANA XS 服务器所处的根路径来使用的。例如包“SAP.TEST”里面包含有应用描述文件，那么当客户端需要调用你的应用时，它需要从“http://<host>:<port>/SAP.TEST/”路径来访问你的应用。因此我们说，应用描述文件需要和你开发的应用包含在同一个

包内才能保证你的应用能够被正确调用。

应用描述文件不包含任何内容，也没有文件名。它是在系统中只会以“.xsapp”作为后缀名的文件出现。

(1) 在“Project Explorer”视图中右击“XSJSHELLOWORLD”项目，选择“New”→“File”(见图 12-18)。

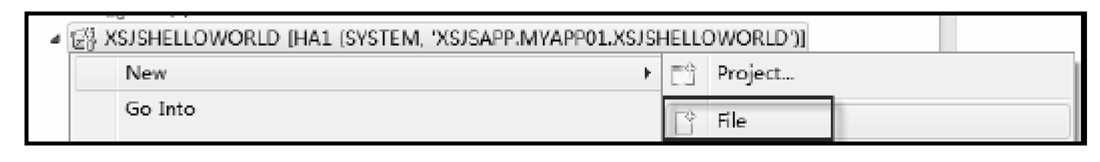


图 12-18

(2) 在弹出的窗口中“File name”字段输入“.xsapp”。注意请不要在这里输入任何文件名，只需输入后缀名即可。输入完成后单击“Finish”按钮确认(见图 12-19)。

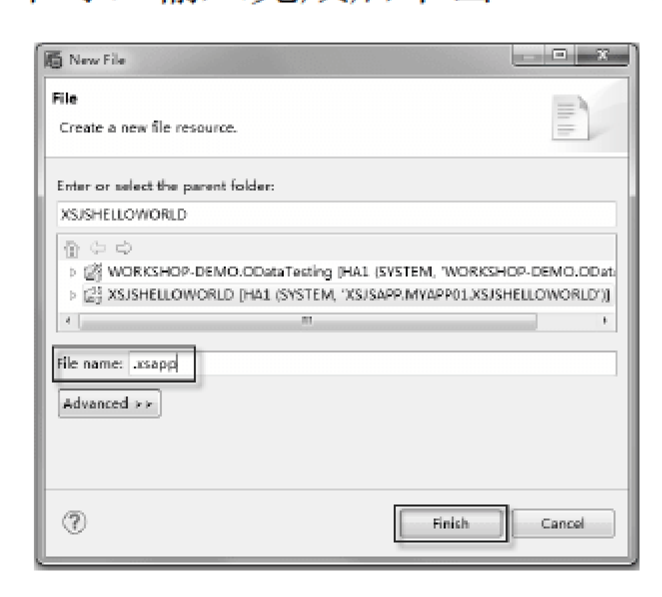


图 12-19

(3) 右击“.xsapp”文件，选择“Team”→“Commit”→“Activate”提交并激活新生成的应用描述文件(见图 12-20)。



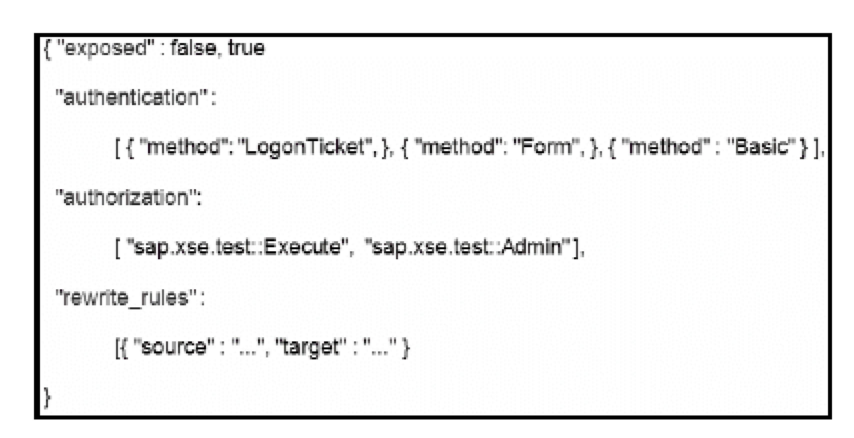
图 12-20

六、创建应用访问文件

应用访问文件(Application Access File)是用来为 SAP HANA XS 服务器应用定义访问控制时使用的。在应用访问文件中，你可以定义哪些用户，需要具备哪些权限可以访问你开发的应用，以及这些用户分别能访问哪些内容。

同应用描述文件一样，应用访问文件同样不需要文件名，在系统中它是以“.xsaccess”为后缀名的文件形式存在的。但是与应用描述文件不同的是，应用访

问文件是需要定义其内容的。图 12-21 所示的是一段定义应用访问文件的常用代码，我们结合这段代码为大家解释代码各部分的组成。



```
{ "exposed" : false, true
  "authentication" :
        [ { "method": "LogonTicket", }, { "method": "Form", }, { "method" : "Basic" } ],
  "authorization":
        [ "sap.xse.test::Execute",  "sap.xse.test::Admin"],
  "rewrite_rules" :
        [{ "source" : "...", "target" : "..." }
}
```

图 12-21

- 数据暴露部分：由 exposed 参数定义，该参数只支持布尔值真(True)或假(False)，其中真代表由此文件控制的应用可以用来做数据暴露，反之假则代表由此文件控制的应用不可以用来做数据暴露。
- 身份鉴定部分：由“authentication”参数定义。在“authentication”参数中，我们可以定义 4 种方式来做用户身份鉴别，这 4 种方式分别为“Null”、“Logon Ticket”、“Form”和“Basic”。
  - “Null”方式即我们不在文件中定义“authentication”参数，意味着任何用户都能访问服务器端的应用。
  - “Logon Ticket”方式支持 SAP Single Sign-On 来使客户端能够访问服务器端应用，这种方式的好处是能够在不需要用户输入用户名和密码的情况下对用户身份进行甄别。但是由于 SAP HANA XS 不支持 SAP Single Sign-On 证书的生成，因此你需要手工配置 SAP Single Sign-On 证书的使用环境。
  - “Form”方式支持通过表单形式来甄别用户身份，即用户需要在表单中填入相应的用户名和密码才可以访问服务器端的应用。
  - “Basic”方式为常见的登录对话框方式，即用户在弹出的对话框内输入用户名和密码来访问服务器端应用。

除了“Null”方式必须单独使用外，其余三种方式在我们定义应用访问文件时既可以单独使用，也可以组合使用来实现多方式身份甄别。例如我们将“Logon Ticket”和“Form”方式组合在一起后，系统在得到用户访问申请时，会首先检查该用户有没有 SAP Single Sign-On 证书，如果有则让用户自动登录；如果没有或证书过了有效期，系统则会要求用户填充登录表单来实现登录功能。这种组合方式在日常开发工作中会经常用到。

- 程序权限部分：由“authorization”参数定义。在“authorization”参数中我们定义用户可以访问服务器端程序所在包裹的权限水平。例如应用所在包裹为“SAP.XSAPP.TEST”，我们定义访问用户对此包裹内的程序只具有执行权限，则可以通过“SAP.XSAPP.TEST::Execute”语句来限定用户权限。而如果我们想给用户更高的权限，则能通过“SAP.XSAPP.TEST:: Admin”语句来赋予访问用户管理员权限。
- URL 路径改写规则部分：由“rewrite_rules”参数定义。“rewrite_rules”参数用于隐藏 URL 中包含系统内部路径的部分。这样一来外部的用户或者搜索引擎将无法得知系统内部路径的具体内容，提高系统的安全性。“rewrite_rules”参数由“源”和“目标”两部分组成，我们引用这个参数时，只需要按照“{"source": "...","target":"..."}”格式定义好需要改写的部分(源)和改写成(目标)即可。

接下来我们具体看看创建应用访问文件的步骤：

(1) 在“Project Explorer”视图中右击“XSJSHELLOWORLD”项目，选择“New”→“File”(见图 12-22)。

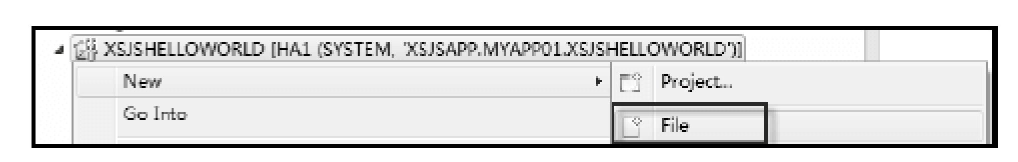


图 12-22

(2) 在弹出的窗口中“File name”字段输入“.xsaccess”。注意请不要在这里输入任何文件名，只需输入后缀名即可。输入完成后单击“Finish”按钮确认(见图 12-23)。

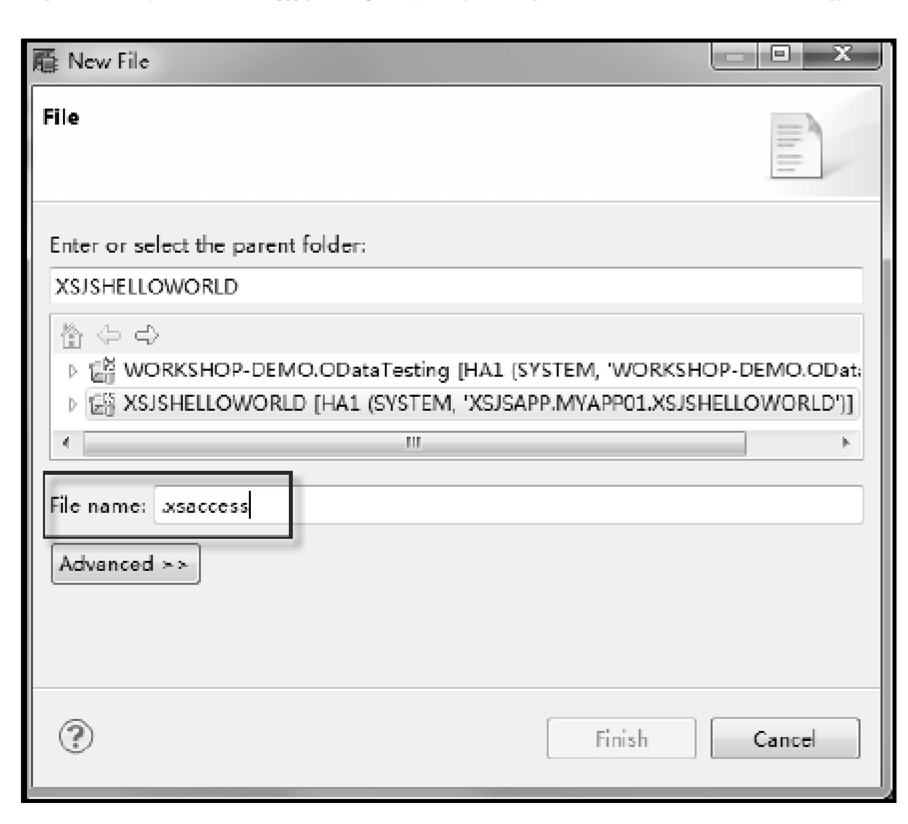


图 12-23

(3) 在编辑区域输入如图 12-24 所示的代码。

```
.xsaccess
{ "exposed":true,
  "authentication":[{ "method": "LogonTicket"}, { "method": "Form"}]
}
```

图 12-24

我们在用户身份甄别时使用了前面介绍的两种方式组合，即系统会首先检测登录用户有没有 SAP Single Sign-On 的证书，如果有即可直接登录，如果没有，则需要在登录表单中输入用户名和密码，下面我们看看运行结果(见图 12-25)。



图 12-25

如图 12-25 所示，由于系统没有检测到发出登录请求的用户有 SAP Single Sign-On 证书，因此系统自动转到表单登录页面来使用户进行登录操作。



七、编写服务器端 JavaScript

好了，到这一步为止我们已经为编写服务器端应用做好了所有准备，是时候开始代码的编写工作了。我们在这一步要做的就是编写一小段代码生成一个最简单的“Hello World”，使得浏览器在访问我们这个应用时能够显示如图 12-26 所示的这个界面。



图 12-26

(1) 在“Project Explorer”视图中右击“XSJSHELLOWORLD”项目，选择

“New”→“Other...”(见图 12-27)。

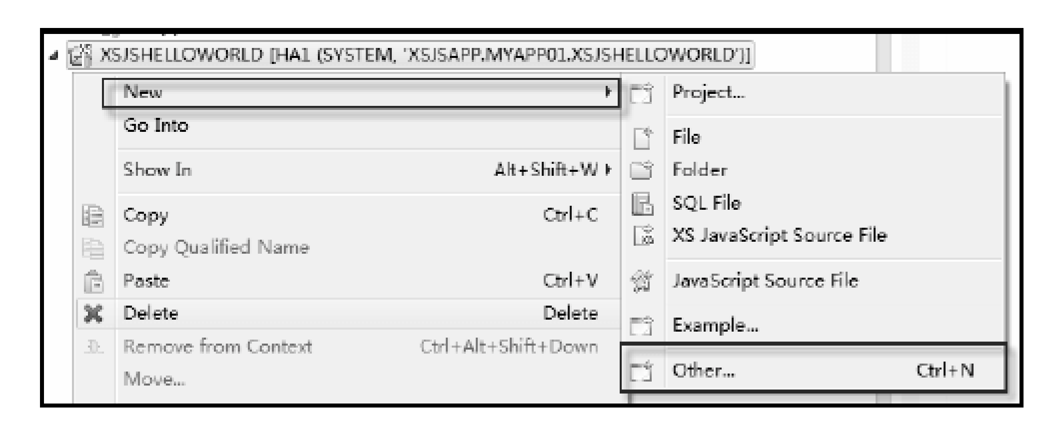


图 12-27

(2) 在弹出窗口中展开“SAP HANA Development”文件夹，选择“XS JavaScript Source File”。单击“Next”按钮继续(见图 12-28)。

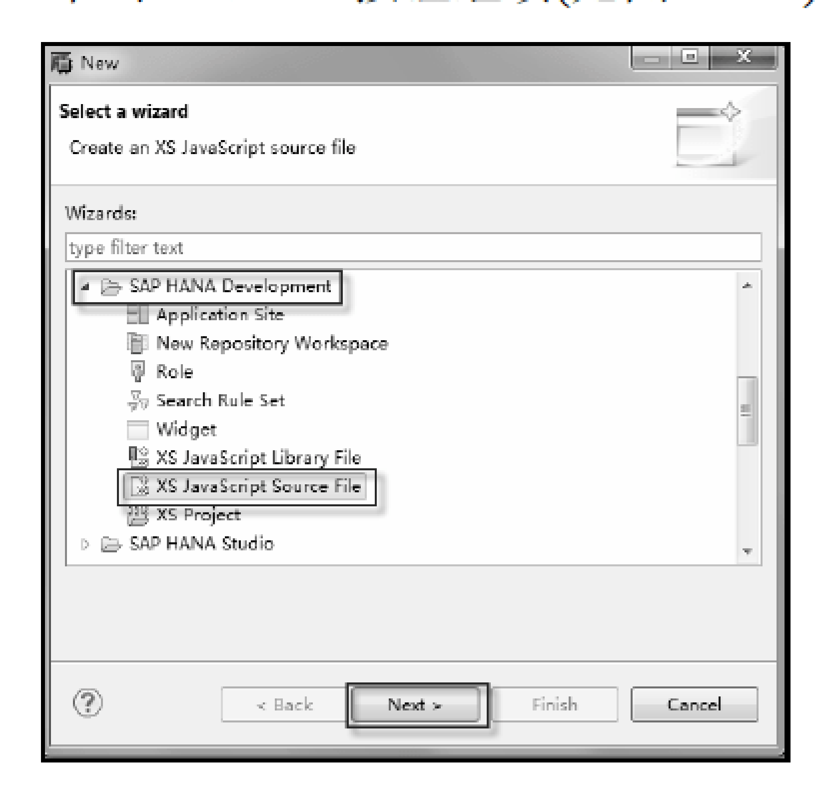


图 12-28

(3) 在“File name”字段输入“MyFirstApps.xsjs”，单击“Finish”按钮确认(见图 12-29)。

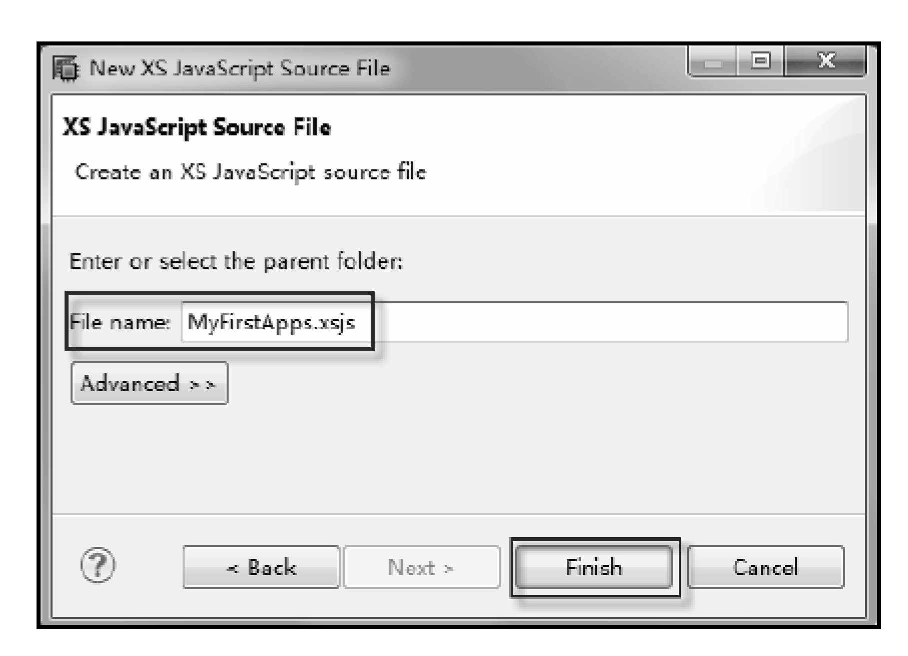


图 12-29



(4) 在编辑区输入如图 12-30 所示的代码。



图 12-30

(5) 右击“MyFirstApps.xsjs”，选择“Team”→“Commit”→ “Activate”提交并激活文件(见图 12-31)。

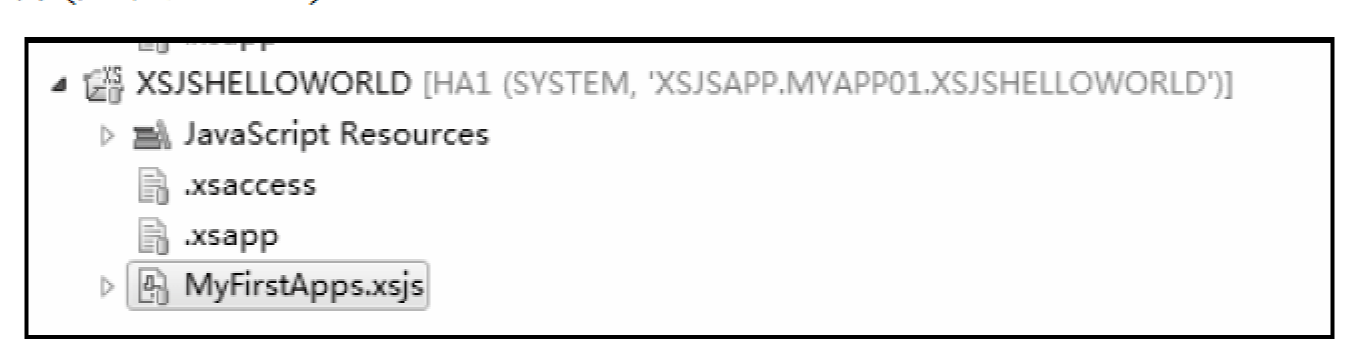


图 12-31

(6) 在浏览器中输入 URL“http://<Your Host>/XSJSAPP/MYAPP01/XSJSHELLOWORLD/MyFirstApps.xsjs”来访问新建的服务器端应用。首先系统会提示你在表单界面输入用户名和登录密码，由何种方式来甄别用户身份是我们在创建“.xsaccess”文件时定义的。

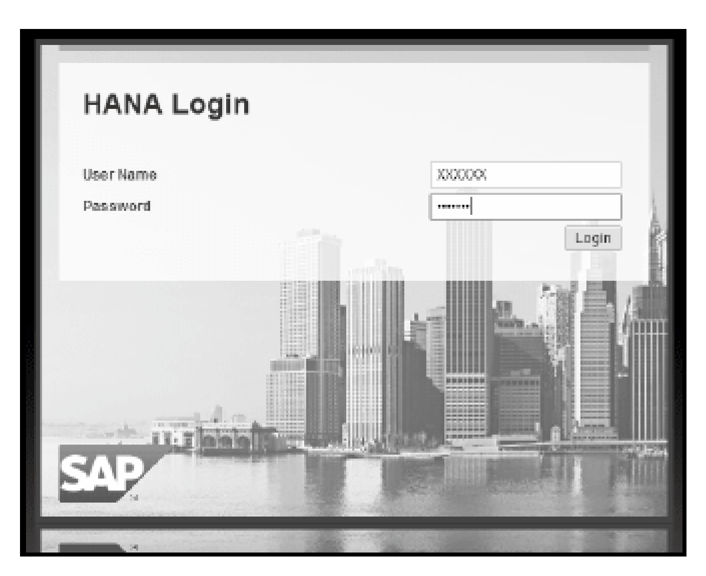


图 12-32

接下来浏览器会取得访问服务器端应用的权限，返回的结果如图 12-33 所示，即“Hello，World！”。



图 12-33



八、暴露数据源数据

前面我们创建了第一个 SAP HANA XS 服务器端应用——“Hello World”。在本小节我们将介绍如何使用 XSJS 代码来编写暴露指定数据源数据的应用。和 OData 服务一样，XSJS 应用也可以通过 HTTP 请求来暴露 SAP HANA 数据库中指定数据源的数据。如果你已经跟随操作流程完成了上一小节中的实例创建，那么接下来你只需要修改“MyFirstApps.xsjs”文件即可，要是你还没有创建这个实例，请按照上小节的流程先创建实例。

接下来我们开始修改“MyFirstApps.xsjs”文件，使之能暴露数据源。

(1) 指定需要暴露的数据源。这里我们指定在第八章建立的“WORKSHOP_JOIN_CUSTOMER”数据表为数据源(见图 12-34)。

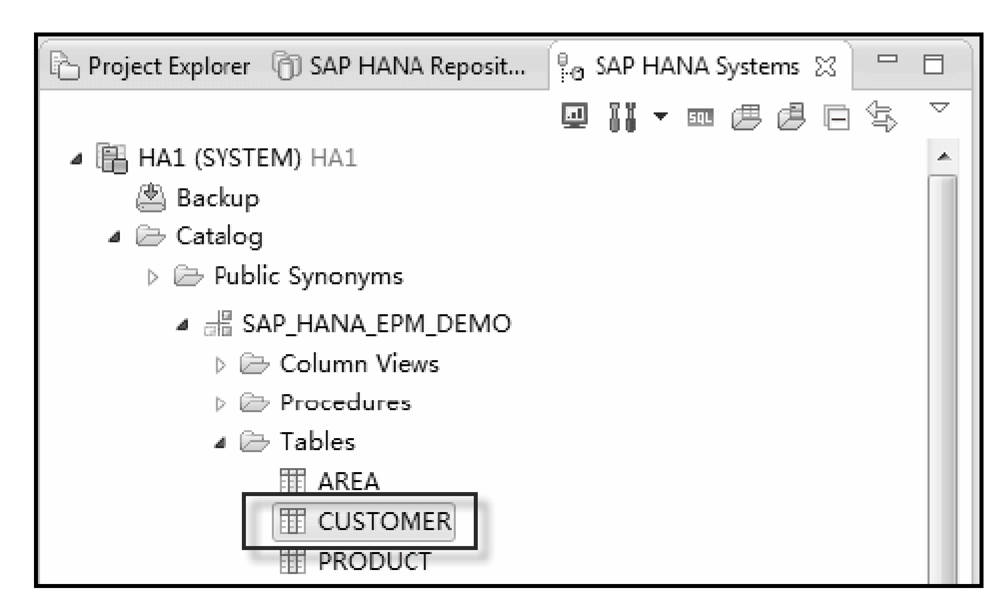


图 12-34

(2) 右击“WORKSHOP_JOIN_CUSTOMER”，选择“Open Content”一项(见图 12-35)。

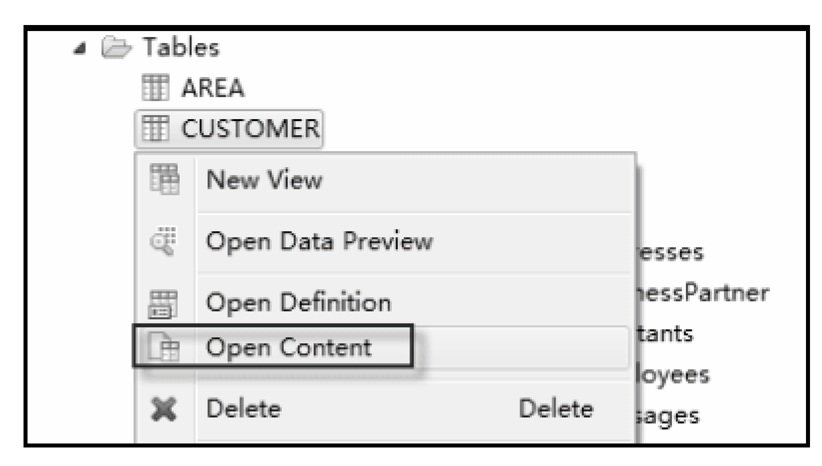
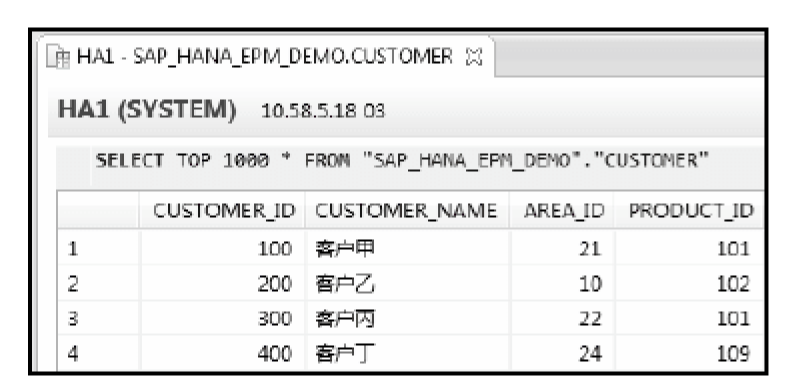


图 12-35

(3) 系统会自动生成一条查询语句来显示该数据表的前 1000 条数据。我们只需要在第 5 步对这条语句进行简单修改既可用于 XSJS 应用(见图 12-36)。


HA1 - SAP_HANA_EPM_DEMO.CUSTOMER

HA1 (SYSTEM) 10.58.5.18 03

SELECT TOP 1000 * FROM "SAP_HANA_EPM_DEMO"."CUSTOMER"

	CUSTOMER_ID	CUSTOMER_NAME	AREA_ID	PRODUCT_ID
1	100	客户甲	21	101
2	200	客户乙	10	102
3	300	客户丙	22	101
4	400	客户丁	24	109

图 12-36

(4) 双击“MyFirstApps.xsjs”文件(见图 12-37)。

Project Explorer　SAP HANA Repositories　SAP HANA Systems
WORKSHOP-DEMO.ODataTesting [HA1 (SYSTEM, 'WORKSHOP-DEMO.ODataTesting')]
XSJSHELLOWORLD [HA1 (SYSTEM, 'XSJSAPP.MYAPP01.XSJSHELLOWORLD')]
JavaScript Resources
.xsaccess
.xsapp
MyFirstApps.xsjs

图 12-37

(5) 在编辑区输入如图 12-38 所示的代码。

*MyFirstApps.xsjs

```
$.response.contentType = "text/html";
var output = "My First App for Data expose <br><br>";
var conn = $.db.getConnection();
var pstmt = conn.prepareStatement( "select TOP 1 * from SAP_HANA_EPM_DEMO.WORKSHOP_JOIN_CUSTOMER" );
var rs = pstmt.executeQuery();
if (!rs.next()) { $.response.setBody( "Failed to retrieve data" );
$.response.status = $.net.http.INTERNAL_SERVER_ERROR;}
else
{ output = output + "查询结果: " + "<br><br>" +"客户ID:"
+rs.getString(1)+";"+"\t"+"客户名称:"+rs.getString(2)+";"+"\t"+"区域代码:"+rs.getString(3)+";"+"\t"
+"产品ID:"+rs.getString(4);}
rs.close();
pstmt.close();
conn.close();
$.response.setBody(output);
```

图 12-38



我们一共在上图这段代码中定义了 4 个变量。变量“Output”负责输出内容的显示。变量“conn”负责声明应用与 SAP HANA 数据库连接。在这里大家能清楚地感觉到，不同于传统的 ODBC 和 JDBC 方式的数据库连接，我们在 XSJS 应用中甚至不需要在声明变量中传递任何参数就能很简单的连接到 SAP HANA 数据库。变量“pstmt”负责声明 SQL 查询，而在本例中我们使用的查询语句就是将第 3 步中的查询语句复制后，将数字“1000”改为“1”，即我们只取数据表第一条数据。变量“rs”则负责声明 SQL 查询的执行。我们使用.getstring()函数来取得数据表返回记录的每列的数值，使用.close()函数来清除各个变量的内容。

将修改好的代码文件提交并激活，在浏览器中输入 URL 调用此应用，可以得到如图 12-39 所示的结果。

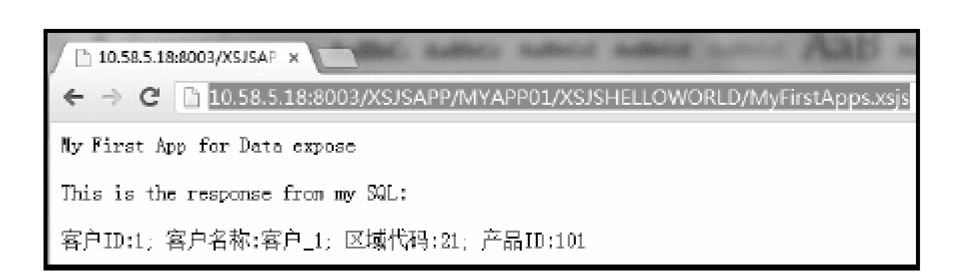


图 12-39

至此，我们已经完成了简单的“Hello World”应用以及暴露简单的数据源应用，接下来我们深入了解关于 SAP HANA XS API 相关的知识。

第二节　应用程序编程接口

SAP HANA XS 提供了一系列的服务器端应用程序编程接口(APIs)以使得应用开发者能够非常灵活地配置其应用来与 SAP HANA 平台做互动。

目前，SAP HANA XS 提供如下三种类型的 API 接口：

- 数据库类 API(Database API)
- 请求处理类 API (Request-Processing API)
- 出站类 API(Outbound API)

一、数据库类 API

通过在应用中引用数据库类 API，你可以使用 SQL 语句来访问 SAP HANA 数据库。例如你可以通过此类 API 声明一个数据库连接来对 SAP HANA 数据库进行提交/回滚变更的操作，或者执行已经存在于服务器端的存储过程(或 SQL 语句)来返回其结果集或者结果集的元数据。接下来我们看几个实例来更好地理解这一类 API。

例 1：数据库查询操作

```
// 创建数据库连接并执行 SQL 查询语句
   var conn = $.db.getConnection();
 var pstmt = conn.prepareStatement( "select * from DUMMY" );
 var rs = pstmt.executeQuery();

 // 将结果集写回载体(网页)
 $.response.contentType = "text/plain";
```

```
if (!rs.next()) {
    $.response.setBody( "Failed to retreive data" );
    $.response.status = $.net.http.INTERNAL_SERVER_ERROR;
} else {
    $.response.setBody("Response: " + rs.getString(1));
}

// 清除变量
rs.close();
pstmt.close();
conn.close();
```

例 2：对数据库做插入操作

```
// 创建数据库连接并执行 SQL 插入语句
var conn = $.db.getConnection();
var st = conn.prepareStatement
("insert into SAP_HANA_EPM_DEMO.AREA values(?,?)");
st.setString(1, '26');
st.setString(2, 'test');
st.execute();
// 对数据库提交插入
conn.commit();

// 清除变量
 st.close();
 conn.close();

// 将结果集写回载体(网页)
$.response.contentType = "text/html";
$.response.setBody("插入数据成功!");
```


本例运行结果如图 12-40 所示。

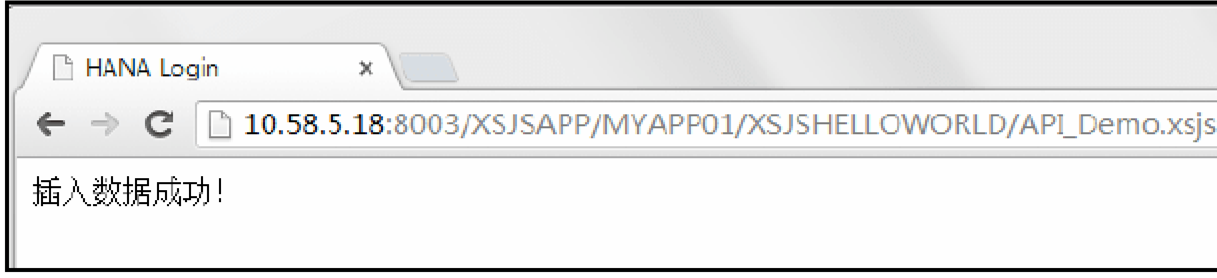


图 12-40

数据表插入新条目后结果如图 12-41 所示。

	AREA_ID	AREA_DESCRIPTION
1	10	北京
2	22	天津
3	21	上海
4	23	重庆
5	25	南京
6	27	武汉
7	26	test

图 12-41

例 3：对数据库做更新操作

```
    // 创建数据库连接并执行 SQL 更新语句
    var conn = $.db.getConnection();
    var st = conn.prepareStatement("UPDATE SAP_HANA_EPM_DEMO.WORKSHOP_
JOIN_AREA
    SET AREA_DESCRIPTION = ? WHERE AREA_ID = ?");
    st.setString(1, '26');
    st.setString(2, 'test_update_demo');
    st.execute();
    // 取得更新的条目数
    var ef_nu=st.executeUpdate();
    // 对数据库提交插入
    conn.commit();

    // 清除变量
    st.close();
    conn.close();
    // 将结果集写回载体(网页)
    $.response.contentType = "text/html";
    $.response.setBody("更新"+"\t"+ef_nu+"\t"+"条数据成功! ");
```

本例运行结果如图 12-42 所示。



图 12-42

数据表更新条目后结果如图 12-43 所示。



	AREA_ID	AREA_DESCRIPTION
1	10	北京
2	22	天津
3	21	上海
4	23	重庆
5	25	南京
6	27	武汉
7	26	test_update_demo

图 12-43

二、请求处理类 API

通过在应用中引用请求处理类 API，你可以对当前 HTTP 请求中的内容进行操控，例如你可以读取 HTTP 请求中的内容，或者将响应信息写到载体上。同样的，我们准备了一些实例来帮助大家更好地理解此类 API。

例 1：显示信息

```
// 显示“Hello World! ”
$.response.contentType = "text/plain";  // 定义输出类型为普通文本格式
$.response.setBody( "Hello, World !"); // 将信息写到载体上
```

运行结果如图 12-44 所示。

图 12-44

例 2：根据输入参数控制显示内容

```
$.import("XSJSAPP.MYAPP01.XSJSHELLOWORLD","demo"); // 导入库文件
var nameParam = $.request.parameters.get("name"); // 定义输入参数
var greeting = $.XSJSAPP.MYAPP01.XSJSHELLOWORLD.demo.greet(nameParam);
$.response.contentType = "text/plain";
$.response.setBody(greeting);
```

运行结果如图 12-45 所示。



图 12-45



三、出站类 API

出站类 API 允许你所创建的应用通过 SAP HANA XS 调用已经定义好的外部的 HTTP 服务。例如你能读取与外部 HTTP 连接端的状态，请求数据并将这些数据体现在你的应用中。

下面这段代码为此类应用的示范代码：

```
var dest = $.net.http.readDestination("inject", "ipsec");
var client = new $.net.http.Client();
var req = new $.web.WebRequest($.net.http.GET, "");
client.request(req, dest);
var response = client.getResponse();
var co = [], he = [];
for(var c in response.cookies) {
    co.push(response.cookies[c]);
}
for(var c in response.headers) {
     he.push(response.headers[c]);
}
var body = undefined;
if(response.body)
    var body = response.body.asString();
$.response.contentType = "application/json";
```

这段代码显示了如何通过“Outbound API”来取得远程 HTTP 终端的数据，包括页面的表头、页面具体内容以及“cookies”。

在这里需要注意的是，我们使用“Outbound API”之前，需要在 SAP HANA XS 里面维护相应远程终端的目的地，即“HTTP Destinations”。“HTTP Destinations”定义了 SAP HANA XS 和远程终端的详细连接信息，并能被任何 SAP HANA XS 应用引用。接下来的小节里面，我们将一步步介绍如何创建“HTTP Destinations”。

四、定义 HTTP Destinations

1. 分配角色

在定义“HTTP Destinations”前，我们需要为做定义的用户分配权限。在 SAP HANA SPS06 版本中，我们可以找到如图 12-46 所示的角色来分配给用户。

sap.hana.xs.admin.roles::HTTPDestAdministrator	_SYS_REPO
sap.hana.xs.admin.roles::HTTPDestViewer	_SYS_REPO
sap.hana.xs.admin.roles::RuntimeConfAdministrator	_SYS_REPO
sap.hana.xs.admin.roles::RuntimeConfViewer	_SYS_REPO
sap.hana.xs.admin.roles::SAMLAdministrator	_SYS_REPO
sap.hana.xs.admin.roles::SAMLViewer	_SYS_REPO
sap.hana.xs.admin.roles::SQLCCAdministrator	_SYS_REPO
sap.hana.xs.admin.roles::SQLCCViewer	_SYS_REPO
sap.hana.xs.admin.roles::TrustStoreAdministrator	_SYS_REPO
sap.hana.xs.admin.roles::TrustStoreViewer	_SYS_REPO

图 12-46

请根据实际需要为用户分配角色，例如角色“sap.hana.xs.admin.roles::HTTPDest Viewer”为只读权限，仅能查看“HTTP Destinations”的定义文件，而“sap.hana.xs.admin.roles::SQLCCAdministrator”角色则能增加/编辑 SQL 连接配置。

2. 创建“HTTP Destinations”定义文件

打开 SAP HANA Studio，切换到“SAP HANA Development”透视图，在“Project Explorer”视图下面找到你需要定义“HTTP Destinations”文件的包，右击调出右键菜单，选择“New”→“File”(见图 12-47)。

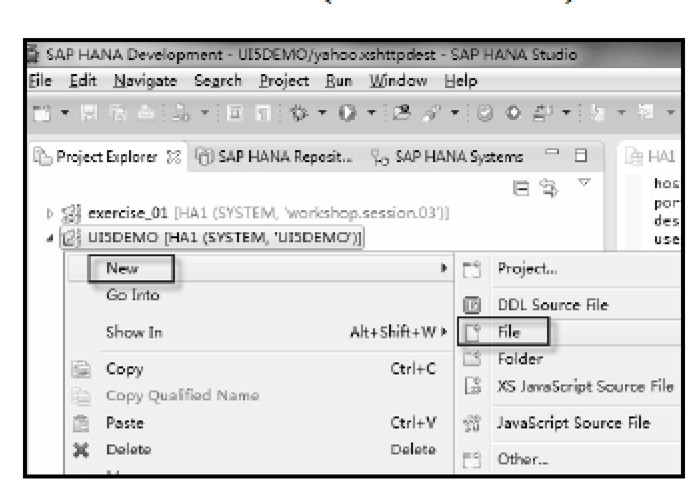
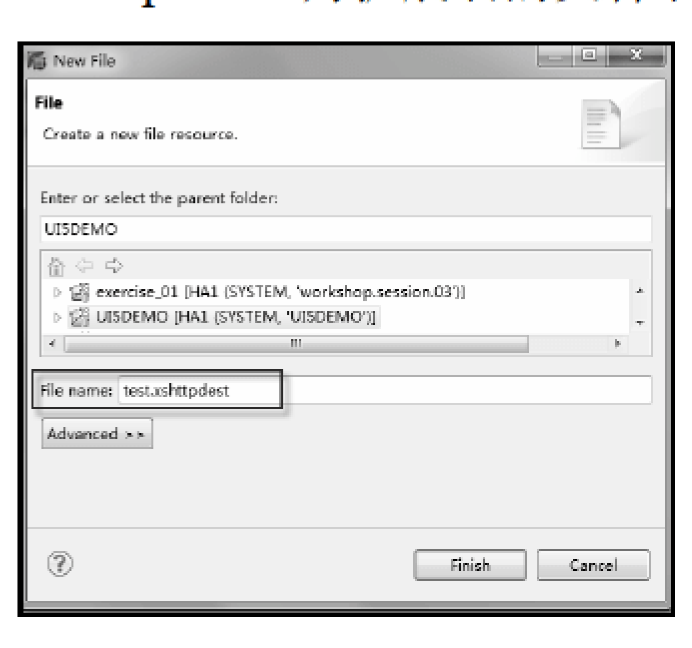


图 12-47

在弹出窗口中输入以“.xshttpdest”为扩展名的文件名，如图 12-48 所示。



12-48



单击“Finish”按钮，在右侧空白处输入如下代码：

```
host = "download.finance.yahoo.com";
port = 80;
description = "my stock-price checker";
useSSL = false;
pathPrefix = "/d/quotes.csv?f=a";
authType = none;
useProxy = false;
proxyHost = "";
proxyPort = 0;
timeout = 0;
```

提交并激活此文件。至此，我们完成了“HTTP Destinations”文件的定义操作。这里我们引用了 Yahoo 的开放 API 来检查股票价格。你也可以引用其他网站开放的 API 来满足你自己应用的需要。有一点需要注意的是，这个“HTTP Destinations”文件一定要和将来需要引用它的 SAP HANA XS 应用文件放在同一个包下面，否则会出现引用不成功的现象。

3. 调用“SAP HANA XS Administration Tool”

“SAP HANA XS Administration Tool”是一个基于网页的在线工具，它能够完成很多诸如 SAP HANA XS 安全配置、证书管理、SAML 配置等工作。在这里我们也可以使用这个工具来进行“HTTP Destinations”的定义操作。

请使用 http://<WebServerHost>:80<SAP HANAinstance>/sap/hana/xs/admin/这个 URL 来调用“SAP HANA XS Administration Tool”，调用成功后我们能找到刚刚定义好的“HTTP Destinations”文件(见图 12-47)。

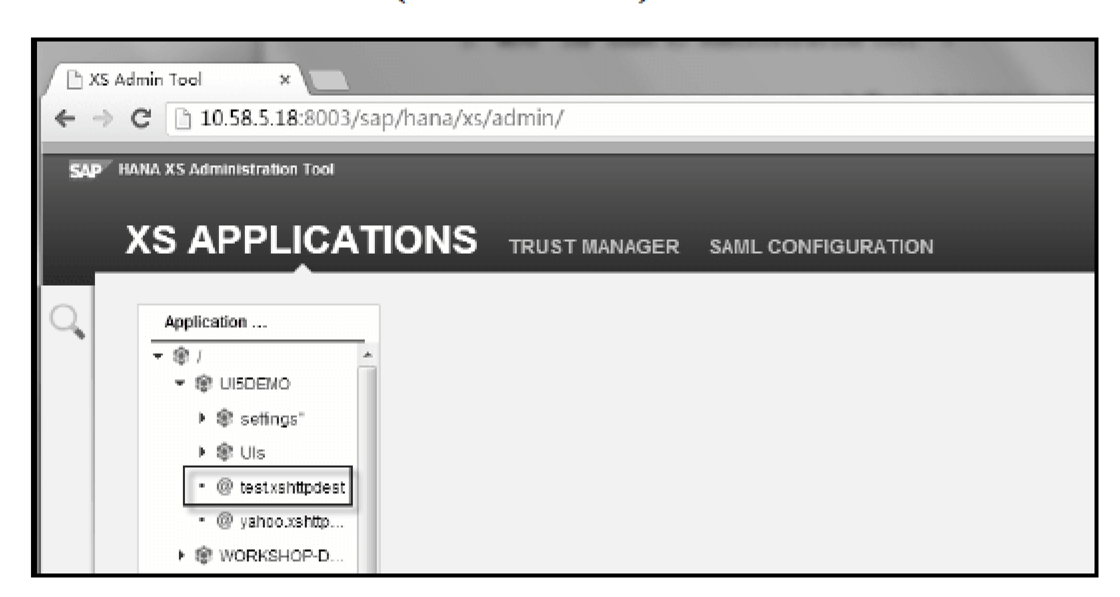


图 12-47

单击此文件，我们能在如图 12-48 所示的区域中进行更加详细的配置。



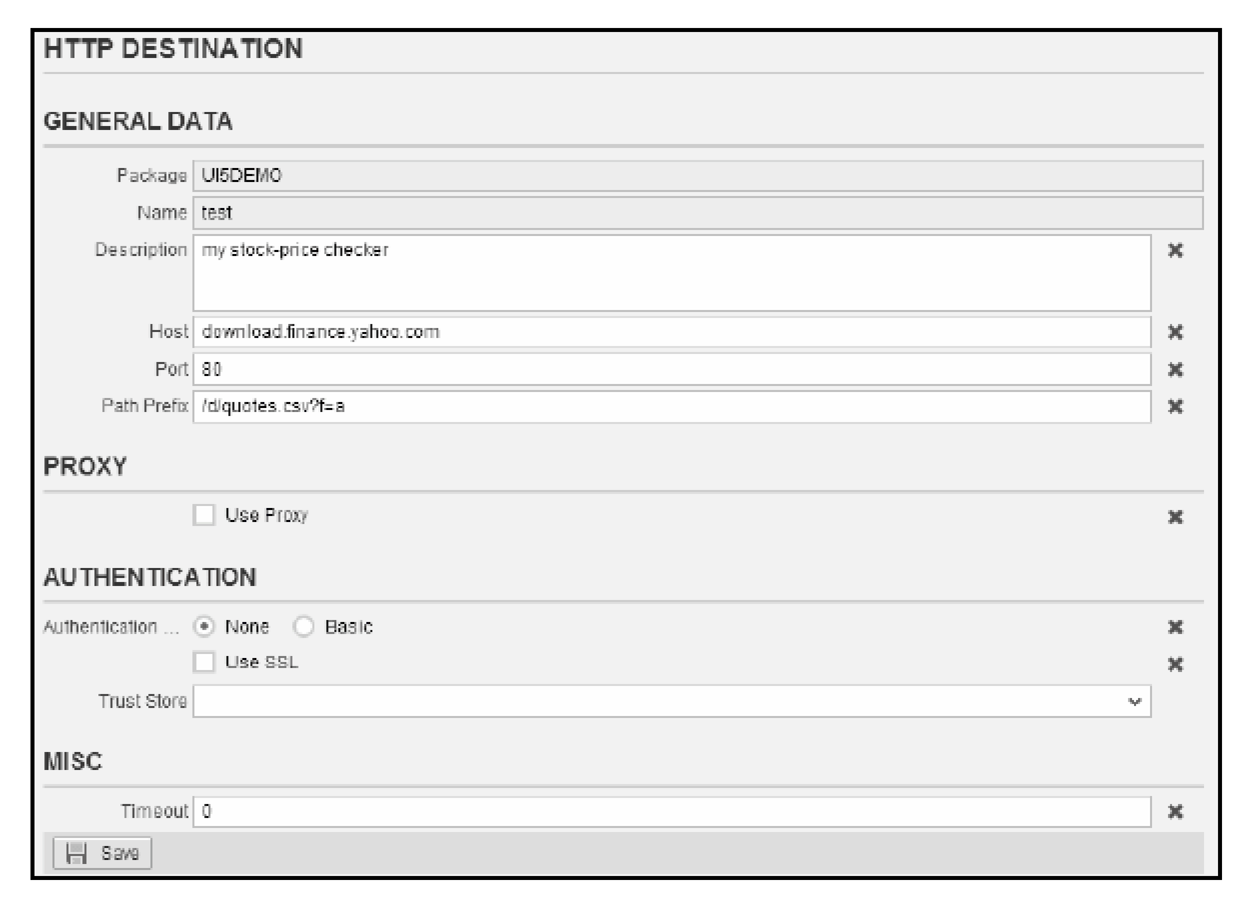


图 12-48

五、使用 HTTP Destinations

在创建好了“HTTP Destinations”文件后，我们新建一个 XSJS 应用来使用它。请按照先前小节介绍的步骤在系统中新建一个 XSJS 应用，并在应用文件中输入如下代码：

```
var stock = $.request.parameters.get("stock");
var amount = $.request.parameters.get("amount");

var dest = $.net.http.readDestination("testApp", "yahoo");
var client = new $.net.http.Client();
var req = new $.web.WebRequest($.net.http.GET, "&s=" + stock);

client.request(req, dest);
var response = client.getResponse();

var co = [], he = [];
for(var c in response.cookies) {
    co.push(response.cookies[c]);
}

for(var c in response.headers) {
     he.push(response.headers[c]);
}
```

```
var body = undefined;
if(response.body)
    var body = response.body.asString();

$.response.contentType = "application/json";

var res = parseInt(response.body.asString()) * amount;

$.response.setBody(amount + " of your " + stock + " are worth: " + res);
```

在这段代码中，我们定义了如下变量：

- 变量<stock>，定义股票名称。
- 变量<amount>，定义需查询的股票数量，例如 100 股。
- 变量<dest>，用来检索“HTTP Destinations”文件数据，例如：host，port，useSSL，等等。
- 变量<client>，用来为出站连接生成客户端。
- 变量<req>，用来生成请求 URL。
- 变量<res>，用来计算股票价格。

在浏览器中调用此 XSJS 应用后，我们能得到如图 12-49 所示的结果。

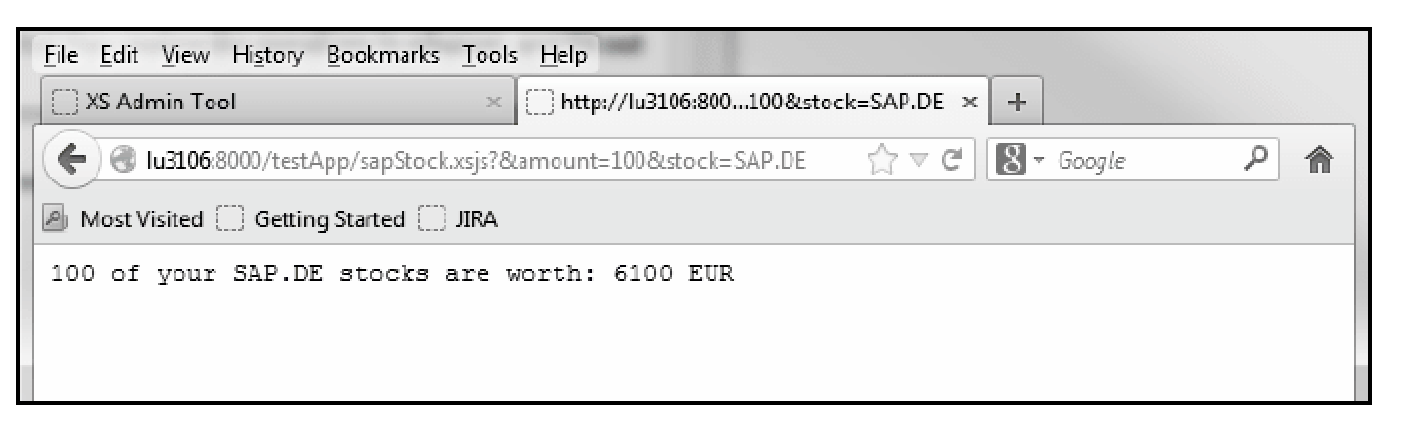


图 12-49

至此我们介绍完了 SAP HANA XS 三大类 API，下面我们继续讲解下 SAP HANA XS 库的概念。

第三节　SAP HANA XSJS 库

有些细心的读者可能已经注意到了我们在第二节第二项下例 2 中使用了“import”类来导入了一个库文件，因此我们仅仅使用了几行代码就实现了根据



URL 的输入参数来动态显示网页内容功能的应用。由此可见 SAP HANA XSJS 库文件的引用能大大简化我们的应用代码，并使得代码逻辑变得更加通俗易懂。那什么是 SAP HANA XS 库文件呢？在实际编程中，某个 XSJS 应用中所包含的各种部件(包括参数、语句、函数等)是不能被其他 XSJS 应用调用的，因此如果我们想在其他 XSJS 应用实现一样的功能，只能把需要的部件再重写一遍或者复制过来。为了节省程序开发者的工作量，提高程序的开发效率，SAP HANA XS 引入了库的概念。SAP HANA XS 库提供了存储应用部件的功能，与此同时它还可以方便地被 SAP HANA XS 应用调用，因此实现了部件的重用。

从程序角度来看，SAP HANA XS 库文件就是一类特殊的 JavaScript 程序，这种程序可以实现某种可重复运行的功能，并可以被其他 JavaScript 应用调用。有一点需要注意的是，虽然我们说 SAP HANA XS 库是一种 JavaScript 程序，但是这种程序只能运行于 SAP HANA XS 服务器端，不能用于其他可运行 JavaScript 的环境中，例如客户端或者浏览器。

一、创建库文件

(1) 打开 SAP HANA Studio，在“Project Explorer”视图右击需创建库文件的项目，在右键菜单中选择“New”→“Other”(见图 12-50)。

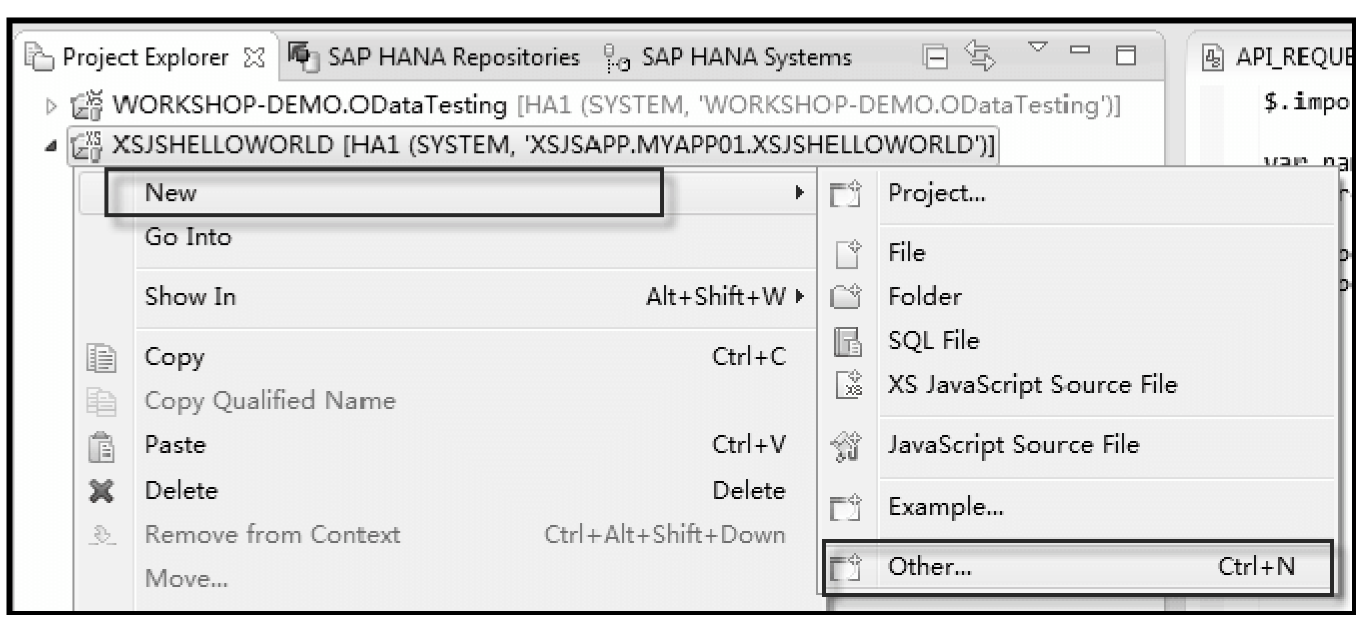


图 12-50

(2) 在弹出的窗口中展开“SAP HANA Development”文件夹，在列表中选中“XS JavaScript Library File”，单击“Next”按钮继续(见图 12-51)。



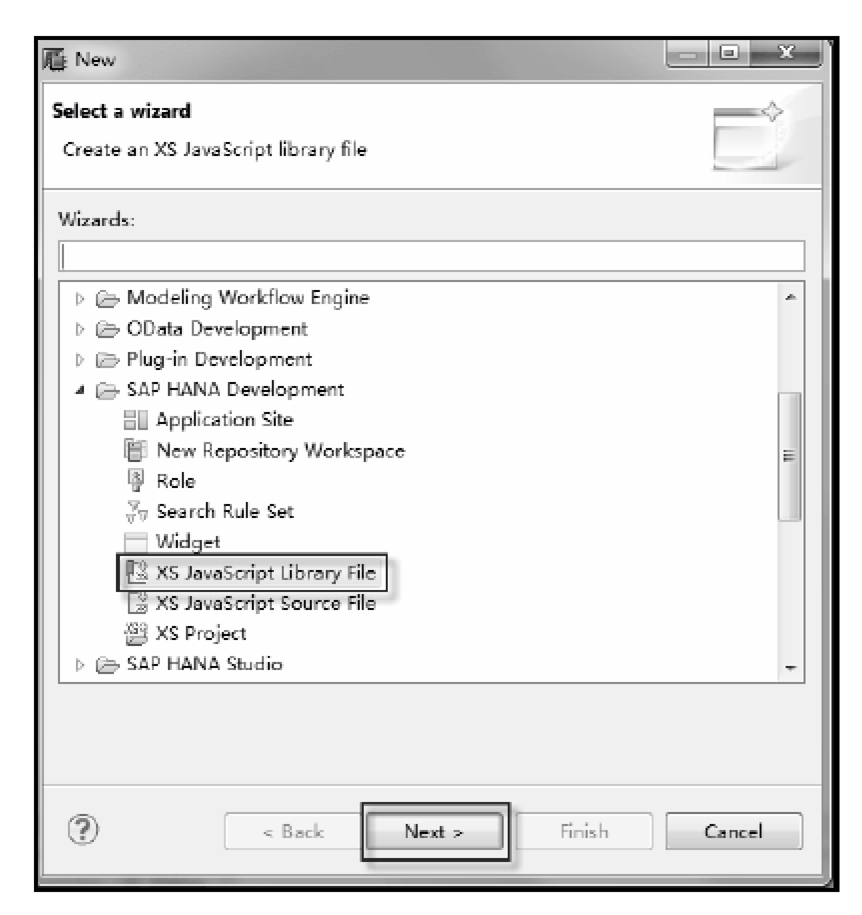


图 12-51

(3) 在下一窗口输入库文件名称，文件名需以“.xsjslib”作为后缀名，单击“Finish”按钮确认(见图 12-52)。

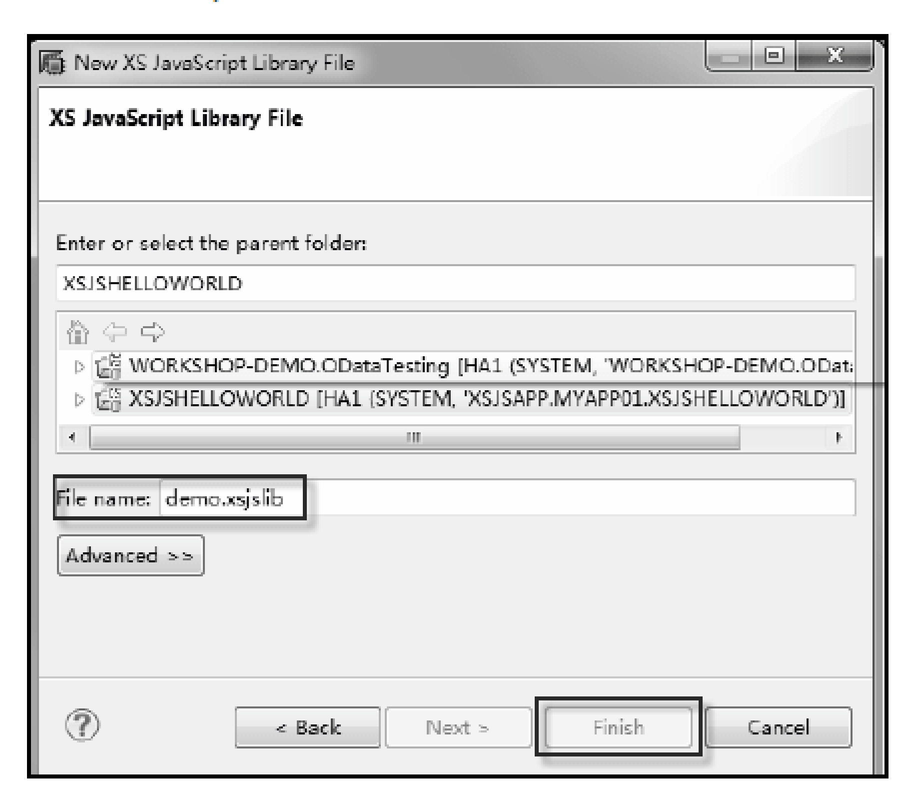


图 12-52

(4) 在编辑区输入如下代码并提交和激活库文件。

```
var greetingSuffix = "Hello, ";

var greetingPrefix = "!";

function greet (name) {
   return greetingSuffix + name + greetingPrefix;
```

```
}

function greetFail (name) {
   return name + " goodbye";
}
```

这是一段很简单的代码，定义了非常通用的应用开始和结束语，我们可以在任何应用的欢迎界面调用这个库文件的“greet”函数来显示欢迎语句，在应用结束时调用“greetFail”函数来结束访问。欢迎和结束时问候的对象则由参数“name”来传递。

二、调用库文件

我们使用“$import”功能来调用库文件，“$import”功能由两个参数组成，即包路径和库文件名称。下面这段代码显示了如何通过“$import”功能调用我们刚才创建的“demo”库文件。

```
$.import("XSJSAPP.MYAPP01.XSJSHELLOWORLD","demo");

var nameParam = $.request.parameters.get("name");
var greeting = $.XSJSAPP.MYAPP01.XSJSHELLOWORLD.demo.greet(nameParam);

$.response.contentType = "text/plain";
$.response.setBody(greeting);
```



将这段代码输入到空白的 XSJS 应用中，提交并激活此应用(见图 12-53)。

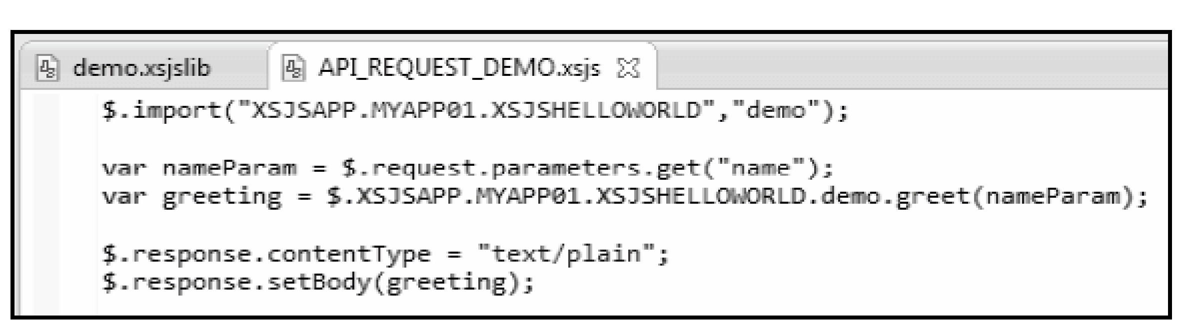


图 12-53

将应用在浏览器中调用(见图 12-54)。



图 12-54

注意“name”参数是通过在 URL 中的赋值语句取得参数值的，而传参的方法我们使用了前面介绍的请求类 API 中的“parameters.get()”方法。接下来我们继续编写一个稍复杂的实例来帮助大家进一步加深对 SAP HANA XS 库的理解。

我们新建一个库文件，将其命名为“arithmetic.xsjslib”并输入如图 12-55 所示的代码。

```
arithmetic.xsjs    arithmetic.xsjslib
var answer; var num=1;

function performArithmetic(oper,num1,num2){

    switch(oper){

    case "add": answer-(num1*num)+(num2*num);
    break;
    case "sub": answer=(num1*num)-(num2*num);
    break;
    case "mul": answer=num1*num2;
    break;
    case "div": answer=num1/num2;
    break;
    default: answer="No Result!Please input necessary parameters";
    }

    return answer;

}
```

图 12-55

稍读下程序大家就知道这是一个通用的进行数学运算的库文件，其中我们需要在调用时输入三个参数，即“算法”、“数字 1”、“数字 2”。返回值“answer”则作为运算结果作为输出参数。下面我们新建一个 XSJS 应用来调用这个库文件(见图 12-56)。

```
arithmetic.xsjs    arithmetic.xsjslib
$.import("UI5DEMO","arithmetic");

var opret = $.request.parameters.get("opreti");
var numb1 = $.request.parameters.get("numb1i");
var numb2 = $.request.parameters.get("numb2i");

var result = $.UI5DEMO.arithmetic.performArithmetic(opret,numb1,numb2);


$.response.contentType = "text/plain";
$.response.setBody(result);
```

图 12-56

我们在程序的第一段首先引入了刚才创建的库文件，接下来在第二段定义了输入参数是由“parameters.get()”方法从 URL 地址中取得。第三段代码定义了由参数“result”取得调用的库文件函数的返回值。我们在第四段代码中将结果输出到网页上。在 URL 中输入如下参数值后我们看看最终结果：

- Opreti = ‘add’，numb1i = ‘6’，numb2i =‘8’ // 执行加法运算(见图 12-57)



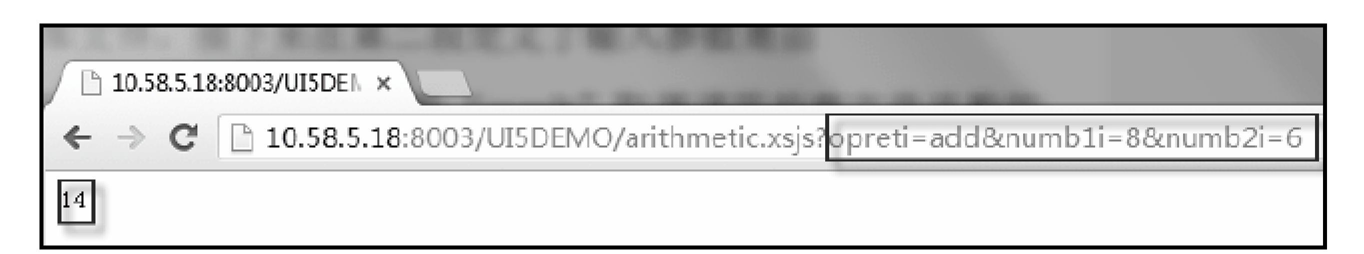


图 12-57

- Opreti = 'mul'，numb1i = '6'，numb2i = '8' // 执行乘法运算(见图 12-58)

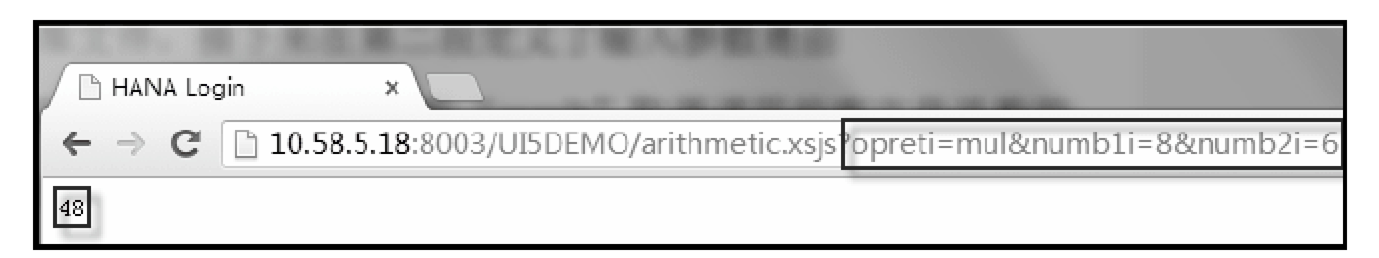


图 12-58

- Opreti = 'div'，numb1i = '9'，numb2i = '3' // 执行乘法运算(见图 12-59)



图 12-59

从这两个实例中我们可以发现，合理运用 SAP HANA XSJS 库可以大大简化代码的复杂度，节省了编程的时间和精力，特别是当许多应用都同时使用一个库文件时，你只需要对这个库文件进行维护和更新，而不用逐一维护其他的应用，这极大地提高了应用发布后维护的效率。



本章小结与练习

通过本章的学习，大家应该能够掌握如何在 SAP HANA Studio 中开发简单的 SAP HANA XS 原生应用，如何通过这些应用来暴露数据源。同时应该对 SAP HANA XS 所支持的 APIs 有了一定的了解。

大体来讲，如果你需要创建一个 SAP HANA XS 应用，你应该遵循如下步骤：

- 检查你所使用的开发用户是否具有相应的开发权限，例如数据库访问权限，开发工具权限等。
- 创建交付单元。

- 创建工作区。
- 创建项目。
- 共享项目并激活。
- 创建“.xsapp”和“.xsaccess”文件。
- 进行项目开发。

练习

1. 请创建“Hello World”应用。
2. 请尝试在你创建的应用中使用 API。
3. 请尝试创建 SAP HANA XS 库文件，并在应用中调用。

第十三章　R 语言在 SAP HANA 中的应用

第一节　R 语言初步接触

一、R 语言简介

什么是 R 语言？R 语言是属于 GNU 系统的一个自由、免费、源代码开放的软件语言，同时它又可以被理解成实现该语言的一个统计计算和制图的优秀工具。它是统计领域广泛使用的诞生于 1980 年左右的 S 语言的一个分支。R 语言是 S 语言的一种实现。S 语言是由 AT&T 贝尔实验室开发的一种用来进行数据探索、统计分析、作图的解释型语言。最初 S 语言的实现版本主要是 S-PLUS。S-PLUS 是一个商业软件，它基于 S 语言，并由 MathSoft 公司的统计科学部进一步完善。后来在 1995 年由新西兰奥克兰大学的罗伯特・杰特曼(Robert Gentleman)和罗斯・艾卡(Ross Ihaka)及其他志愿人员开发了一个 R 系统。R 语言的使用与 S-PLUS 有很多类似之处，两个软件有一定的兼容性。S-PLUS 的使用手册，只要经过不多的修改就能成为 R 语言的使用手册。换句话说：R 语言，是 S-PLUS 的一个“克隆”，但 R 语言是免费的。R 系统首页的图形如图 13-1 所示。

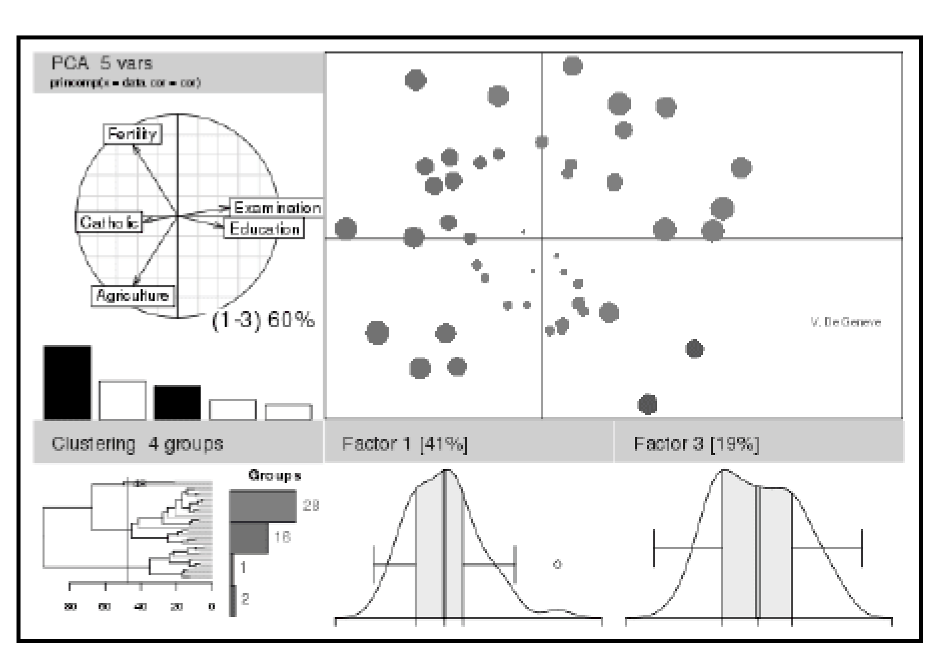


图 13-1



R 语言是一套完整的数据处理、计算和制图软件系统。其功能包括：

- 数据存储和处理系统；
- 数组运算工具(其向量、矩阵运算方面功能尤其强大)；
- 完整连贯的统计分析工具；
- 优秀的统计制图功能；
- 简便而强大的编程语言：可操纵数据的输入和输入，可实现分支、循环，用户可自定义功能。

与其说 R 语言是一种统计软件，还不如说 R 语言是一种数学计算的环境，因为 R 语言并不仅仅提供若干统计程序。使用者只需指定数据库和若干参数便可进行一个统计分析。R 语言的思想是：它可以提供一些集成的统计工具，但更大量的是它提供各种数学计算、统计计算的函数，从而使使用者能灵活机动地进行数据分析，甚至创造出符合需要的新的统计计算方法。R 语言内建多种统计学及数字分析功能。R 语言的功能也可以透过安装套件(Packages，用户撰写的功能)增强。增加的功能有特殊的统计技术、绘图功能，以及编程界面和数据输出/输入功能。这些软件包是由 R 语言、LaTeX、Java 及最常用 C 语言和 Fortran 撰写的。下载的执行档版本会连同一批核心功能的软件包，而根据 CRAN 记录有上千种不同的软件包。其中有几款较为常用，例如用于经济计量、财经分析、人文科学研究以及人工智能。因为 S 语言的血缘，R 语言比其他统计学或数学专用的编程语言有更强的物件导向(面向对象程序设计)功能。此外虽然 R语言主要用于统计分析或者开发统计相关的软体，但也有人用作矩阵计算。其分析速度可媲美 GNU Octave 甚至商业软件 MATLAB。

R语言具有很多优点:



- 免费：R语言是完全免费的。
- 开放：R语言是 S 语言的开放源代码做的版本，你可以将 S-PLUS 的程序直接放入 R语言中执行。
- 占有率高：SAS 是最普遍被使用的软件，但是在学术界最普及的软件是 R 语言与 S 语言，尤其是在统计的期刊中，常常可以看到 R 语言的踪迹。
- 跨平台：R 语言可以在各种平台上运作，包含 Windows、Macintosh、Linux 等数十种平台。
- 扩展性强：R 语言语言是一种程序语言，使用者可以自行撰写适合自己的分析程序。
- 互动性：传统的统计分析软件，是将所有的统计分析过程一次做完，产生报表。而 R 语言可以互动式地一步一步处理，使用者可以按照每一步的结果而决定下一步如何处理。

目前 R 语言在高校非常流行，特别是随着这几年互联网的发展。R 语言在一些大公司的运用得到的实践，例如：国外的 Google、Linkedin、Facebook 等，国内一些大型互联网公司也在开始使用 R 语言。同时互联网版权的意识增强，也促使了 R 语言在互联网的发展。当然 R 语言在很多领域都有很广泛的运用。

二、R 语言环境简介

最简单的使用 R 语言的方法莫过于在一个桌面系统的图形工作站使用运行 R 语言了。 在本书的后文中，我们会讲述在 SuseLinux 上安装 R 语言的过程。现在，为了方便读者了解熟悉 R 语言，我们先从 Windows 开始来熟悉了解 R 语言的语法。

1. 安装 R 语言

(1) 访问 R 语言官方网址 http://www.r-project.org。
(2) 进入下载页下载 Windows 安装包。
(3) 运行 R-win.exe，完成安装并运行。
(4) 运行后，我们可以看到如图 13-2 所示的命令行界面。

> 虽然 R 语言是支持中文的，但是在 R 语言中，大部分包(package)的作者是以英语为母语或者首要语言，并没有对中文字符考虑过多。因此，我们建议在 R 语言中使用全英文环境。

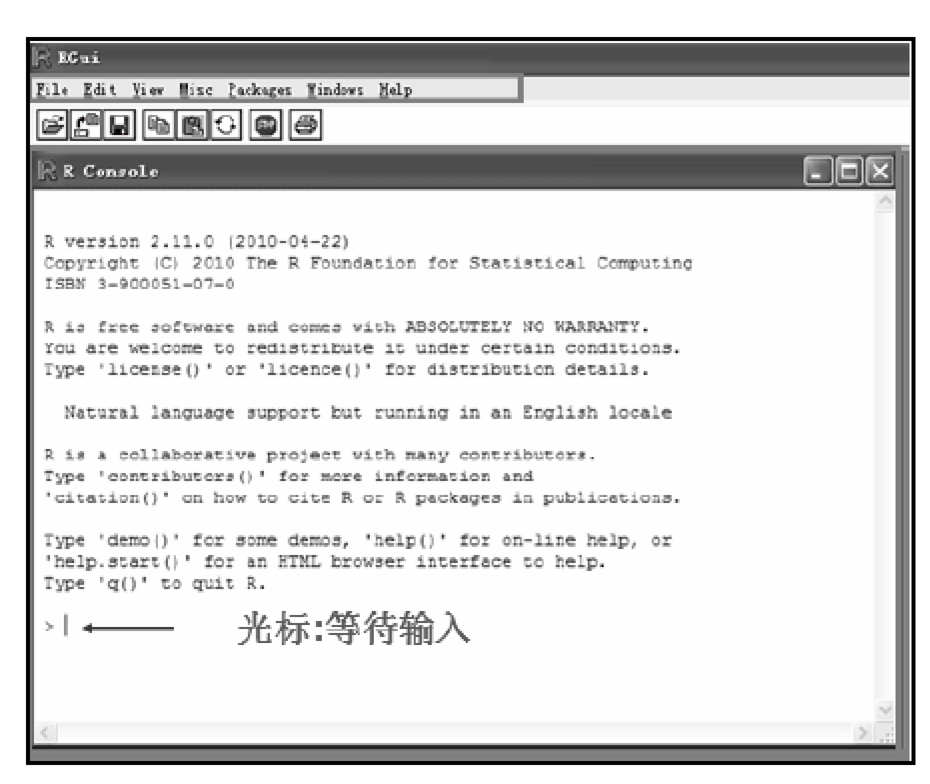


图 13-2



2. R 语言简单命令

(1) R 语言是一种交互式的语言，所以在 R 语言环境中当我们看到“>”时表示它正在等待输入。输入完命令后，则通常 R语言会直接输出当前的结果。

(2) 一个对象可以通过赋值操作来产生，R 语言中的赋值符号一般是由一个尖括号与一个负号组成的箭头形标志(“<-”)。输入下面命令完成赋值操作：

```
>  x<-15; y<-20
```

(3) 直接输入变量名显示变量。可以看到 R语言会立即返回输出，通常在输出行会以“[1]”开始。

```
>x
[1]  15
>y
[1]  20
```

(4) 显示内存中所有变量。

```
>  ls ()
[1]  "x""y"
>  ls.str ()
x :  num 15
y :  num 20
```

(5) 删除内存中的变量。

```
>  rm(x)
>  ls()
[1]  "y"
```

(6) 获取帮助。我们会看到帮助文档会以网页形式打开。

```
>  help.start()
Starting httpd help server …done
If nothing happens,you should open
'http://127.0.0.1:15428/doc/html/index.html'  yourself
```

恭喜，你已经完成一些最简单又常用的 R语言操作了！



三、R 程序包

在接触了 R 语言的最简单的一些命令之后，我们来了解另外一个 R 语言的基础：R

程序包。

R 程序包是多个函数的集合，具有详细的说明和示例。Windows 下的 R 程序包是经过编译的 zip 包。每个程序包包含 R 语言函数、数据、帮助文件、描述文件等。R 程序包是 R 语言功能扩展，特定的分析功能，需要用相应的程序包实现。一些常用的包如：

- Cluster：聚类分析
- FD：功能多样性分析
- Graphics：绘图
- Lattice：栅格图
- Maptools：空间对象读取和处理
- SP：空间数据处理
- Stats：R 统计学包

CRAN 提供了每个包的源代码和编译好的程序包。以 vegan 包为例，CRAN 提供了不同的包给不同平台：

- Package source: vegan_1.17-2.tar.gz
- MacOS X binary: vegan_1.17-2.tgz
- Windows binary: vegan_1.17-2.zip

安装程序包步骤如下:

1. 使用函数 install.packages()

如果已经连接到互联网，在括号中输入要安装的程序包名称，选择镜像后，程序将自动下载并安装程序包。此方法的另外一个好处是，当程序包之间有依赖关系时，系统会自动下载。

例如：要安装 cluster 包，在控制台中输入：

```
install.packages ("cluster")
```

安装完之后，我们可以看到如下类似信息：

```
>install.packages("cluster")
Warning in install.packages("cluster"):
  'lib="C:/Program Files/R/R-2.15.1/library"'is not writable
---Please select a CRAN mirror for use in this session---
trying  URL  'http://mirrors.ustc.edu.cn/CRAN/bin/windows/contrib/2.15/
cluster_1.s
```




```
Content type 'application/zip'length 498742 bytes (487 Kb)
opened URL
downloaded 487 Kb
package 'cluster' successfully unpacked and MD5 sums checked
```

2. 安装本地 zip 包(该功能只能在 R GUI 中使用)

另外一个方法是，我们可以直接在 CRAN 的网站上下载 R 程序包至本地，然后在系统菜单中选择“Packages”→“install packages from local files”来完成程序包安装。

导入 R 程序包如图 13-3 所示。

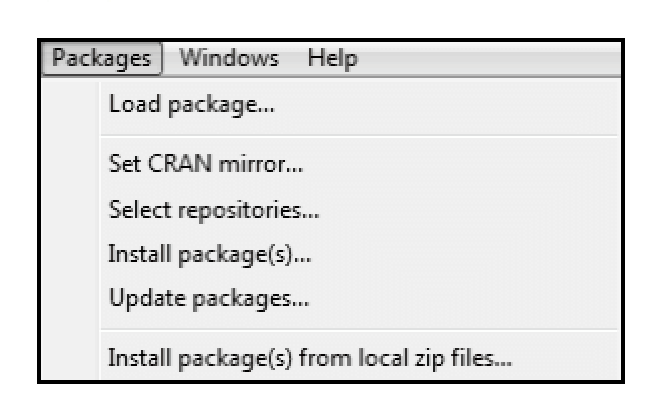


图 13-3

在安装完程序包之后，并不意味着我们就可以直接使用了。为此，我们还需要做第二个步骤：导入程序包。这个可以通过 library 命令来实现。

3. 在 R 控制台中，输入命令：

```
library("cluster")
```

通常如果没有返回信息或者范围信息没有提示出错，则该程序包已经被成功导入。

(1) 使用 sessionInfo 命令可以查看所有已被导入至会话的包(见图 13-4)。

```
> sessionInfo()
R version 2.15.1 (2012-06-22)
Platform: x86_64-pc-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] cluster_1.14.4      DAAG_1.16           survival_2.36-14
 [4] boot_1.3-4          randomForest_4.6-7  rpart_3.1-53
 [7] latticeExtra_0.6-26 RColorBrewer_1.0-5  lattice_0.20-23
[10] MASS_7.3-18

loaded via a namespace (and not attached):
[1] grid_2.15.1  tools_2.15.1
```

图 13-4



(2) 导入程序包成功后，程序包中的函数用法与 R 语言内置函数用法就完全一样了。

第二节　R 语言基础

在本节中，我们将简单地学习一下 R 语言的基本语法。事实上关于 R 语言的各种学习资料，我们已经有非常多的渠道可以得到。因此，在本节中，我们只会简单介绍在本书中需要应用到的一些 R 语言的必备知识。若想对 R 语言有进一步深入了解，可以参考官方网站 http://cran.r-project.org/other-docs.html 上的中文翻译文档。

一、R 语言数据结构

在 R 语言中，我们一直都在与对象(object)打交道。在 R 语言中有以下一些类型:

- 基础类型
  - 实数型(real)：整数(integer)、单精度(single)、双精度(double)；
  - 虚数型(complex)：如 9+11i；
  - 字符型(character，string)：如"hello"(单双引号都行)；
  - 逻辑型(logical)：TRUE，FALSE(简写 T，F)；
  - 函数(function)；
  - 表达式(expression)。
- 结构化数据
  - 向量(vector)：一列数值或字符；
  - 矩阵(matrix)：m 行×n 列(各列之间类型都相同)；
  - 数据框(data frame)：类似矩阵，但每一列的数据类型可以不同；
  - 数组(array)：多维度(不是多变量)；
  - 列表(list)：有诸多成员杂合在一起，这些成员可以是任意类型，甚至是 list 本身(及其灵活的数据类型)；
  - 因子(factor)：分类变量；
  - 时间序列(ts)：时间序列数据。



使用变量的时候要特别注意，R 语言对变量的大小写敏感！



二、R 语言产生数据

在统计这一块，产生示例数据来为模型验证是相当重要且频繁的操作。我们可以从外界(数据库，文本文件)等直接导入数据，也可以在对数据质量要求不高时或者仅作测试时自己手动产生一些实例数据。下面产生数据的命令都可以直接在 R 语言命令行中输入运行。

1. 产生基本数据

(1) 产生一个 1～10 的规则整数序列：

```
> x<-1:10
> x
  []  1  2  3  4  5  6  7  8  9  10
```

(2) 通过下列命令查看“：”与“- “的优先级：

```
>1:10-1
  []  1  2  3  4  5  6  7  8  9
```

(3) 生成等差的实数序列：

```
>seq(1,5,0.5)
[]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0
```

(4) 重复相同对象：

```
>rep(1:3,3)
  []  1  2  3  1  2  3  1  2  3
```

(5) 通过 c 函数直接输入数值：

```
> x<-c(1,3,2,9,9,9,8); x
  []  1  3  2  9  9  9  8
```

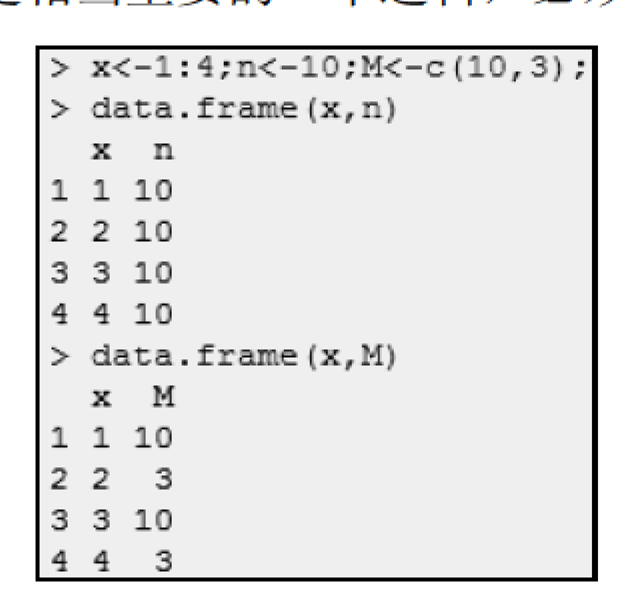
2. 产生结构化数据

(1) 一个矩阵实际上是有一个附加属性(维数 dim)的向量，维数即为一个长度为 2 的向量，用来指定矩阵的行数和列数。一个矩阵可以用函数 matrix 来创建：

```
> matrix (1:6,2,3)
     [,1] [,2] [,3]
[1,]   1    2    5
[2,]   2   4    6
```



(2) 另外一种创建表的方法是使用数据框(data.frame)。数据框中的响亮必须有相同的长度，如果其中有一个比其他的短，它将“循环“整数次(以使得其长度与其他向量相同)。这在 R 语言中是相当重要的一个逻辑，必须牢记(见图 13-5)。



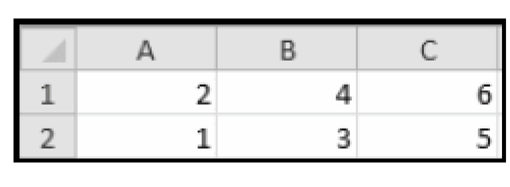
```
> x<-1:4;n<-10;M<-c(10,3);
> data.frame(x,n)
  x  n
1 1 10
2 2 10
3 3 10
4 4 10
> data.frame(x,M)
  x  M
1 1 10
2 2  3
3 3 10
4 4  3
```

图 13-5

3. 读取外部数据

(1) 数据量较大时用 read.table 函数从外部 txt 文件读取。

(2) 如图 13-6 所示，将 Excel 中的数据另存为.txt 格式(制表符间隔)或.csv 格式。



	A	B	C
1	2	4	6
2	1	3	5

图 13-6

(3) 用 read.table()或 read.csv()函数将数据读入 R 工作空间，并赋值给一个对象(见图 13-7)。

```
> x<-read.table("c:/Users/Public/TEST.CSV",header=FALSE)
> x
     V1
1 2,4,6
2 1,3,5
```

图 13-7

三、下标的使用

在 R 中，另外一个较有特色的地方就是对下标的使用。对任何向量，矩阵或者是数据框，我们都可以通过灵活的下标使用来实现我们特殊目的。例如：

(1) 数据量较大时用 read.table 函数从外部 txt 文件读取(见图 13-8)。



```
> x<-1:10
> x[x>5]
[1]  6  7  8  9 10
> x[x > 5 & x < 9]
[1] 6 7 8
```

图 13-8

(2) 对矩阵做下标操作(见图 13-9)。

```
> x = matrix(1:20, 4, 5)
> x[x>=2 & x < 16]
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

图 13-9

(3) 批量赋值(见图 13-10)。

```
> x[x>=2 & x < 16] = NA
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   NA   NA   NA   17
[2,]   NA   NA   NA   NA   18
[3,]   NA   NA   NA   NA   19
[4,]   NA   NA   NA   16   20
```

图 13-10

四、常用数学和统计函数

下面列出的，是在 R 语言中比较常用的一些数学和统计函数：

- 最大值 max()，最小值 min()，均值 mean()，标准差 sd()，方差 var()，相关系数 cor()，求和 sum()，积 prod()，中位数 median()，分位数 quantile()，对数 log()，指数 exp()，排列 factorial()，组合 choose()，四舍五入 round()，向下取整 floor()，向上取整 ceiling()，总结 summary()；
- 累加 cumsum()，秩 rank()，排序 sort()，倒序 rev()，矩阵转置 t()，逆矩阵 solve()，特征根 eigen()；
- 关于统计分布的“四大金刚”：pnorm()，dnorm()，qnorm()，rnorm()。(p，d，q，r+分布名称分别构成：分布函数值、密度函数值、分位数、随机数，如 pf()表示 F 分布函数值，runif()表示产生均匀分布的随机数)抽样 sample()；
- 线性模型 lm()，广义线性模型 glm()，t 检验 t.test()。



五、程序流控制

对于 R 语言中程序流控制，主要由条件语句和循环两种：

1. 条件语句

- 作用：避免除零或负数的对数等数学问题
  - 形式 1：

    if(条件)表达式 1else 表达式 2

  - 形式 2(常优于形式 1)：

    ifelse(条件，yes，no)

举例如图 13-11 所示。

```
> x=c(6:-4);x
 [1]  6  5  4  3  2  1  0 -1 -2 -3 -4
> sqrt(ifelse(x >= 0, x, NA))
 [1] 2.449490 2.236068 2.000000 1.732051 1.414214 1.000000 0.000000       NA
 [9]       NA       NA       NA
```

图 13-11

2. 循环语句

```
for (变量  in  向量)   表达式
while(条件)  表达式
```

两者通常可以转换，举例如图 13-12 所示。

```
> for (i in 1:5) print (1:i)
[1] 1
[1] 1 2
[1] 1 2 3
[1] 1 2 3 4
[1] 1 2 3 4 5
```

图 13-12

六、R 脚本实例

至此，我们有了数据结构，了解了 R 语言的一些基本规则，知道了怎么控制我们的程序，那么，编程的问题基本已经解决。接下来，让我们通过一个 R 语言的实例，来看 R 语言是如何解决实际生活中的相关问题的。

需求：假想这样一个例子，你正在图书馆枯坐，一位陌生美女主动过来和你搭



讪，并要求和你一起玩个数学游戏。美女提议：让我们各自亮出硬币的一面，或正或反。如果我们都是正面，那么我给你 3 元，如果我们都是反面，我给你 1 元，剩下的情况你给我 2 元就可以了。在这种情况下，该不该和这位姑娘玩这个游戏呢？

答案：很多人的第一感觉是正反的组合有 4 种等概率的情况，出现其中两种情况我要付出 2＋2＝4 元，出现另外两种情况可以得到 3＋1＝4 元，不亏不赚，这是个公平的游戏嘛。那就玩呗，更何况人家是美女。

很遗憾，事实上这是一个比较经典的数学陷阱问题。在这里已经不是概率问题了。注意这道题目，正反面是我们自己可以决定的。这个时候，如果我们按随机概率亮出正反面，而对方只需以 1/3 的概率出正面，2/3 的概率出反面，每 6 次游戏你就就会输一元。

也许你会说，那我也可以调整自己的策略。这是对的。当然对任何一个游戏，玩家们都会想方设法让自己的利益最大化。这时你会发现，游戏的局势已经变得出乎意料地复杂。那么，如何通过 R 语言来迅速的解决这个问题呢？

其实最简单的理解方式是设自己出正面的概率是 x，美女出正面的概率是 y。那么每次你的收益为

$$f(x, y)=3xy-2x(1-y)-2(1-x)y+(1-x)(1-y)$$

其中 x，y 大于等于 0，小于等于 1，然后我们在 R 上通过画图统计可以看到你的收益分布。在 R 语言命令行中输入如图 13-13 所示的命令。

```
> library(rgl)
> x<-rep(seq(0,1,0.01),times=100)
> y<-rep(seq(0,1,0.01),each=100)
> rate <- function(x,y) {
+ z<- 3*x*y - 2*x*(1-y) - 2*(1-x)*y + (1-x)*(1-y)
+ return(z)
+ }
> rate(0.5,0.5)
[1] 0
> rate(0.5,0.333333)
[1] -0.166667
> z<-rate(x,y)
> plot3d(x,y,z,col=rainbow(1000))
```

图 13-13

我们可以看到你的收益分布(见图 13-14)。



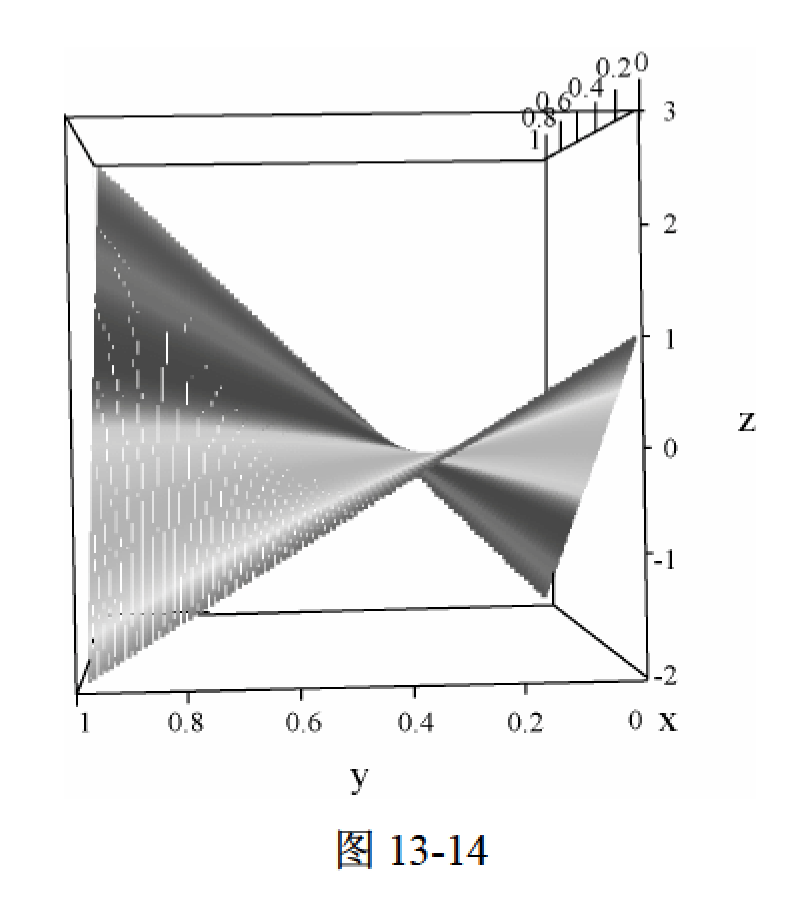


图 13-14

此时，你可以看到，在 y 值靠近 0.4 的地方，无论你的策略如何，你的收益永远是负的。估算 y 值为 0.35 与 0.375，继续输入如下代码可以看到你的预期收益(见图 13-15)：

```
> y <-rate(x,0.35)
> plot (x,y)
>
```

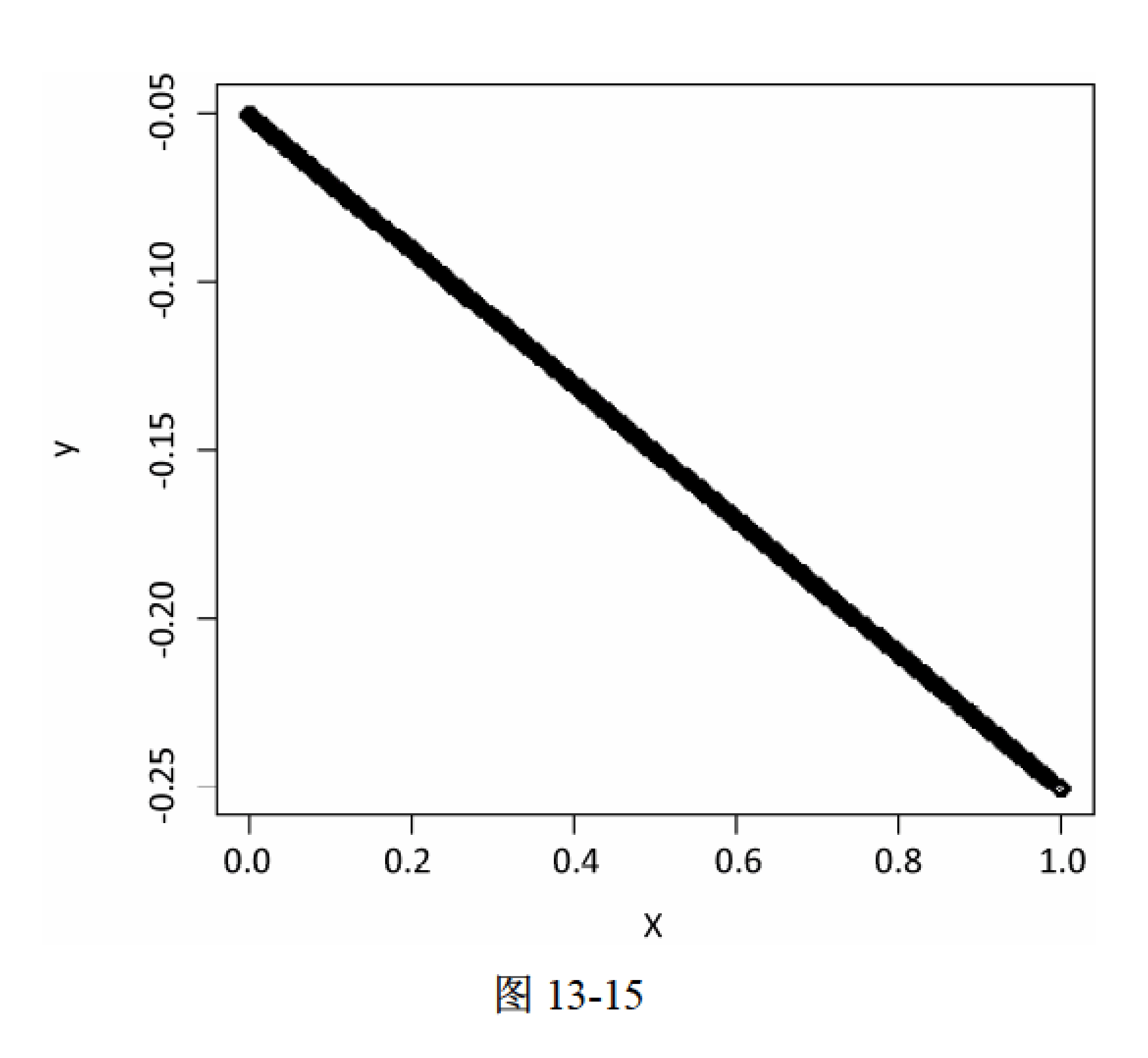


图 13-15

从图可见，当美女以 35%的正面概率来玩这个游戏时，无论我们采取何种策略，我们都不能获胜。有兴趣的读者还可以输入当概率为 37.5%时的情况。事实上，



在数学中，这是一个很著名的纳什均衡问题。而我们通过 R 语言，则较轻松地得到了自己的答案。

第三节 R 语言与 SAP HANA 集成配置

在了解完 R 语言的一些基础之后，让我们把目光移回到我们的主题——SAP HANA 上来。R 语言能为 SAP HANA 带来什么？在了解完这么多 R 语言的功能之后，我们可以很自然地想到：HANA 的速度与 R 语言丰富的函数库相结合，正是我们这个大数据的年代数据分析的绝佳工具。

由于 R 语言是一种开源语言并且基于 GPL 许可，因此，SAP 并不随 SAP HANA 提供 R 语言的环境，并且 SAP 并不提供对 R 语言的支持。为实现 R 语言与 SAP HANA 的集成，我们需要下载 R 语言并且事先做好相应配置。虽然 R 语言有针对不同操作系统的各种版本，但现在 SAP HANA 只支持安装在 SUSE Linux 上的 R 语言实例。另外，我们需要 Rserve， 它是一个 TCP/IP 服务器。它允许不同的编程语言(如 Java、C++)可以不需要做任何 R 语言的初始化或者连接 R 语言的类库来实现相应 R 语言的功能。当一切都配置完毕，我们所达到的目的是能够使 R 语言代码在 SAP HANA 的环境中运行。

一、安装 R 语言



为实现 R 语言与 SAP HANA 集成，我们必须由源代码安装 R 语言。为实现安全性，SAP 推荐另外创建一个专门的 R 语言用户来执行任何与 R 语言的操作。在本书中，为了简单起见，我们会使用超级用户 ROOT 来完成所有操作。另外，在本书撰写的时候，SAP 只成功测试与 R 2.15 版本集成。对 R 语言的最新版本并没有完全测试。读者可以参考 http://help.sap.com/hana/SAP_HANA_R_Integration_Guide_en.pdf 来获取 SAP HANA 支持的 R 语言最新版本。

在安装 R 语言之前，请检查以确保以下包已经被安装:

- readline-devel
- xorg-x11-devel
- gcc-fortran

SUSE 中详细安装 R 语言的步骤如下：

(1) 在 CRAN 官方网站下载 R 语言的源安装包。

(2) 将包解压缩到某一文件夹。

(3) 打开 SUSE 的控制台，进入该文件夹，执行命令：

```
./configure  -enable-R-shlib
```

(4) 开始编译：

```
make clean
make
make install
```

(5) 通常 R 语言会安装在文件夹/usr/local/bin 中，成功安装后，可以直接在控制台中输入 R 语言来检查 R 语言是否已成功安装(见图 13-16)。

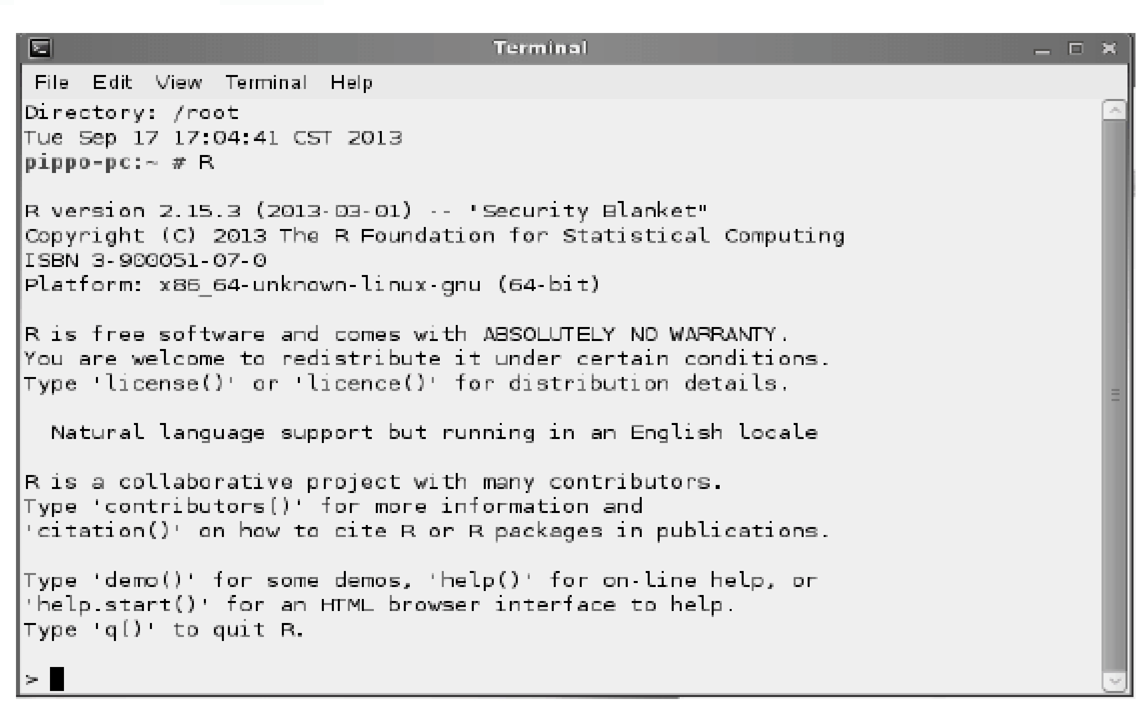


图 13-16

二、安装 Rserve

Rserve 是一个基于 TCP/IP 协议的，允许 R 语言与其他语言通信的 C/S 结构的程序，支持 C/C++、Java、PHP、Python、Ruby、Nodejs 等。它相当于 SAP HANA 与 R 系统中间的连接器，提供远程连接，认证，数据/文件传输等功能。安装 Rserve 的步骤如下：

(1) 从 http://www.rforge.net/Rserve/中下载 Rserve。

(2) 以 root 用户登录 SUSE Linux 并且在控制台输入如下命令：

```
R
install.packages("/PATH/TO/YOUR/Rserve.tar.gz",  repos = NULL)
```

```
library("Rserve") #如果成功， 则不会有任何输出
```

(3) 安装成功后，需要配置 Rserve。请使用具有 root 权限的用户登录系统并创建文件/etc/Rserv.conf，然后输入以下内容：

```
maxinbuf 10000000
Maxsendbuf 0
remote enable
```

这里“10000000”只是一个例子，我们推荐使用内存大小(以字节为单位)/2048 为数值在这里设置，并且假设我们使用 ruser 来启动 Rserve，我们需要赋予 ruser 访问该文件的权限，可以通过如下命令实现：

```
chown -R ruser /etc/Rserv.conf
chmod u+rx /etc/Rserv.conf
```

(4) 使用如下命令来启动 Rserve 服务：

```
R CMD Rserve --RS-port PORT --no-save --RS-encoding "utf8"
```

这里的端口(PORT)需要和 SAP HANA 的端口保持一致，一般设为 3XX20，XX 是 SAP HANA 的实例号。选项 no-save 说明当 R 被停止时，并不会把 R 运行时数据存入文件系统，以减少对系统资源的占用。

(5) 目前 Rserve 并不支持自启动服务。不过我们可以通过使用 crontab 来实现类似的功能。创建一个可执行文件内容如下：

```
cd /usr/local/bin
pgrep -u root -f "Rserve --RS-port <PORT> --no-save" || R CMD
Rserve --RS-port <PORT> --no-save
```



(6) 使用 crontab 脚本将该可执行文件设为开机自动运行。

三、配置 SAP HANA 参数

你需要在 SAP HANA Studio 中配置相关参数来最终完成与 R 语言的集成。所有与 R 语言相关的参数都是在“indexserver.ini”文件的 calcengine 部分完成的。具体操作如下：

(1) 在 SAP HANA Studio 中，双击你的系统节点进入管理界面(见图 13-17)。

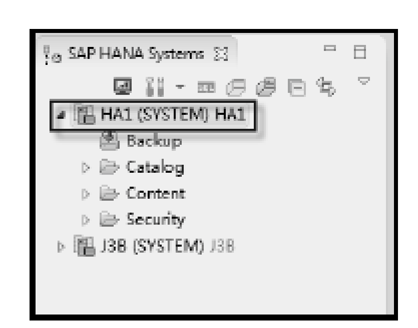


图 13-17

(2) 在右侧选择“Configuration”标签(见图 13-18)。

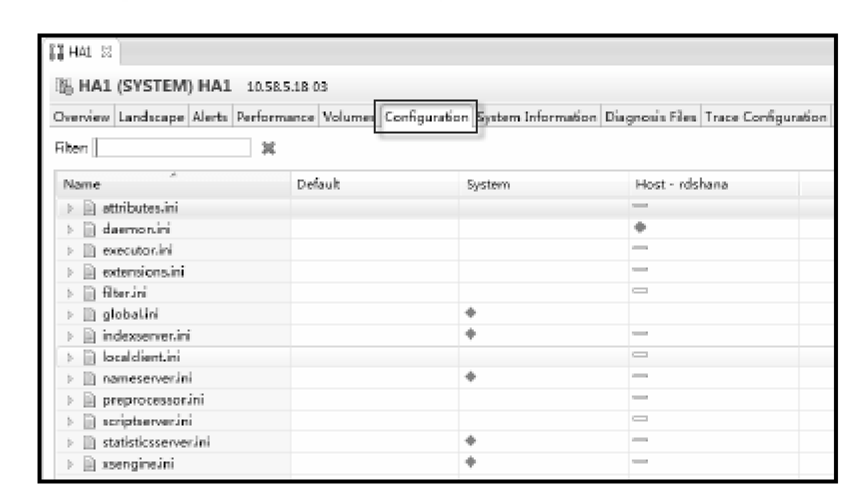


图 13-18

(3) 选择展开“indexserver.ini”节点(见图 13-19)。

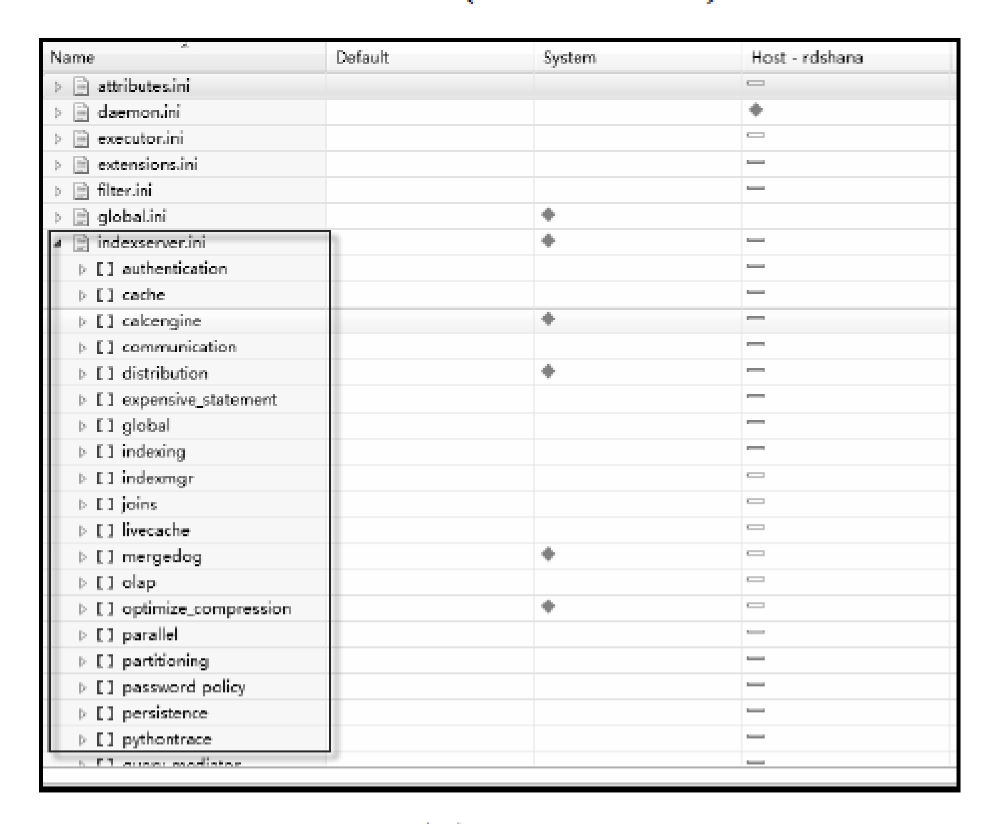


图 13-19

(4) 单击展开“calcengine”(见图 13-20)。

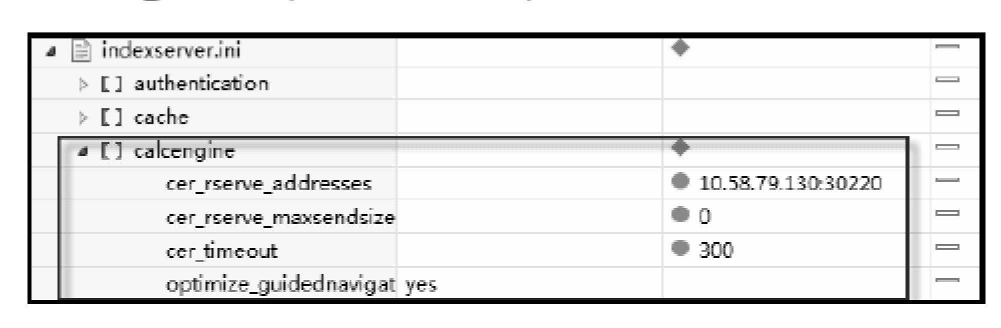


图 13-20



(5) 在 calcengine 下面可以添加如下参数：

- Cer_timeout：设置连接超时，单位为秒，默认为 300。该参数相当重要，如果你希望 R 语言运行时间超过 5 分钟你应该修改这个参数。
- Cer_rserve_addresses：R 语言主机列表，当有多个 R 语言服务器时，可以通过如下格式设置：Host1:Port1、Host2:Port2 等多个服务器可以实现高可用性。
- Cer_rserve_maxsendsize：用于表示从 R 语言到 SAP HANA 中传递结果数据大小的最大值(千字节为单位)。其默认值为 0，表示没有限制，如果结果数据的大小超过限制，则传输中止，抛出错误。

至此，SAP HANA 与 R 语言的集成配置已全部完成.

四、测试 HANA 与 R 语言连接

在本小节中，我们可以简单的在 SAP HANA Studio 中创建一个存储过程来测试。

(1) 在 SAP HANA Studio SQL 控制台中， 输入如下代码创建测试表。

```
DROP TABLE TEST_TABLE;
CREATE TABLE TEST_TABLE (NUMBER INTEGER);
```

(2) 插入测试值：

```
INSERT INTO TEST_TABLE VALUES (2);
INSERT INTO TEST_TABLE VALUES (3);
INSERT INTO TEST_TABLE VALUES (5);
INSERT INTO TEST_TABLE VALUES (7);
INSERT INTO TEST_TABLE VALUES (11);
INSERT INTO TEST_TABLE VALUES (13);
INSERT INTO TEST_TABLE VALUES (19);
```



(3) 创建输出结果表：

```
DROP TABLE TEST_SQL;
CREATE TABLE TEST_SQL (NUMBER INTEGER);
```

(4) 创建 R 存储过程：

```
DROP PROCEDURE TEST_R;
CREATE PROCEDURE TEST_R(IN input1 TEST_TABLE,  OUT result TEST_SQL)
LANGUAGE RLANG AS
BEGIN
```

```
        result <- as.data.frame(input1$NUMBER^2);
        names(result) <- c("NUMBER");
END;
```



为创建 R 存储过程，你需要有系统特权 CREATE R SCRIPT。

(5) 调用 R 存储过程来完成测试：

```
CALL TEST_R (TEST_TABLE, TEST_SQL) WITH OVERVIEW;
SELECT * FROM TEST_SQL;
```

(6) 如果返回结果，则说明数据已经通过 R 执行并返回至 HANA。

第四节　R 应用场景——嵌入式 R 代码

在 SAP HANA 中我们可以有两种方式来实现与 R 集成。

- R 语言单独应用：数据通过与 SAP HANA 的快速交互机制来读入，这种方式比较适合在 R 语言环境中使用 R 语言图形界面的能力来进行交互的开发过程。
- 嵌入式 R 代码：SAP HANA 数据库允许 R 代码通过 HANA 的查询执行计划来处理。这种场景比较适合基于 HANA 的应用需要使用 R 语言的特定统计功能。

其中，第一种方式我们可以理解为传统的 R 语言与数据库集成方式。我们甚至可以通过已有的 RJDBC 包来实现 R 语言与 SAP HANA 的连接。但是如果对 R 有一定的了解，我们会发现在 R 语言中大量的数据结构(vector 等)都是以列为存储结构的，而这与 SAP HANA 的列存储结构完美匹配。因此，SAP 提供了一个单独的 RHANA 包来实现高性能的 R 语言与 SAP HANA 数据通信(RJDBC 是以行为存储单元传输的)。遗憾的是，在本书发稿之时，RHANA 包并没有正式发布，因此我们暂不对第一种方式进行阐述。

对于第二种集成方式，在 SAP HANA 环境中使用 R 语言，实际上我们通过上节中的小例子已对这种方式有一定了解。它通过在 SAP HANA 中创建专门的 R 存储过程，然后对任何该存储过程的调用，SAP HANA 此时会把相应的请求传递给外

部 R 环境来处理。这样做的好处是，可以让开发人员一起发布完整的项目 SQL 与 R 代码，而不必在不同的环境中(见图 13-21)。

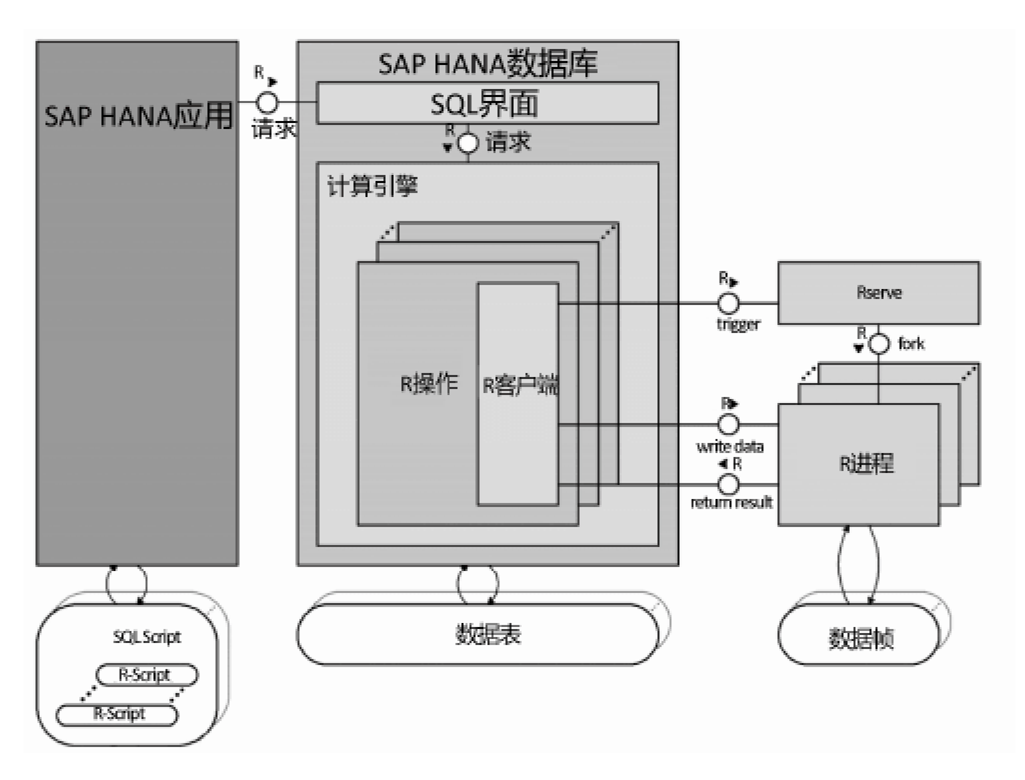


图 13-13

在图 13-21 中我们可以看到有三个主要组件：SAP HANA 应用，SAP HANA 数据库以及 R 环境。当开始执行 R 操作时，计算引擎的 R 客户端发出一个请求，通过 Rserve 在 R 服务器创建一个单独的 R 进程。HANA R 客户端非常高效地传送 R 代码以及输入数据到 R 服务器。当所有 R 操作都结束后，结果会被传回计算引擎。在下节中我们会通过另外一个更详细的例子来了解 R 语言的应用。

一、R 存储过程语法



创建 R 存储过程(R Procedure)与我们前文中讲述的一般存储过程的方法一致，唯一需要注意的地方就是 LANGUAGE 这里需要使用 RLANG，如下所示：

```
CREATE PROCEDURE <proc_name> [(<parameter_clause>)]
LANGUAGE RLANG
AS
 BEGIN
     纯 R 脚本
 END
```

二、追踪 R 存储过程

由于 R 语句是在外部执行的，为了能够看到 R 存储过程的执行时间(包括执行时间与数据传送时间)，我们需要在 SAP HANA 中更改追踪级别，这可以通过 SAP

HANA Studio 来修改 indexserver.ini 参数：

(1) 在 HANA Studio 中，双击你的系统节点进入管理界面。

(2) 在右侧选择“Configuration”标签。

(3) 选择展开“indexserver.ini”节点(见图 13-22)。

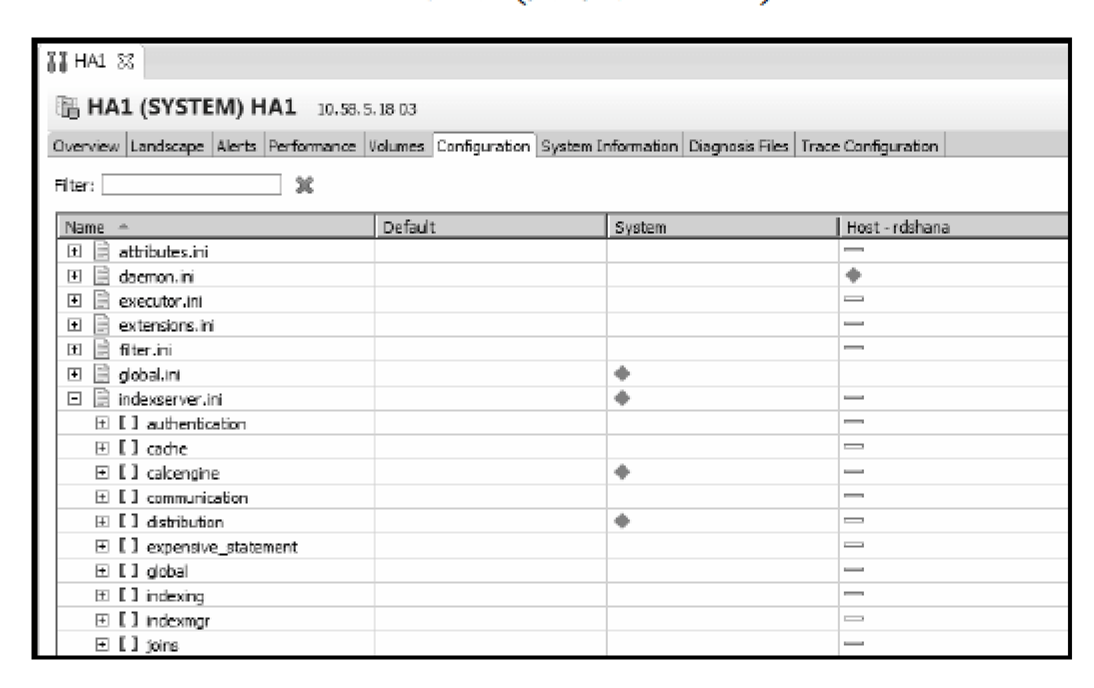


图 13-22

(4) 展开“Trace”节点。

(5) 鼠标右击“Trace”节点并选择“Add Parameter…”一项(见图 13-23)。

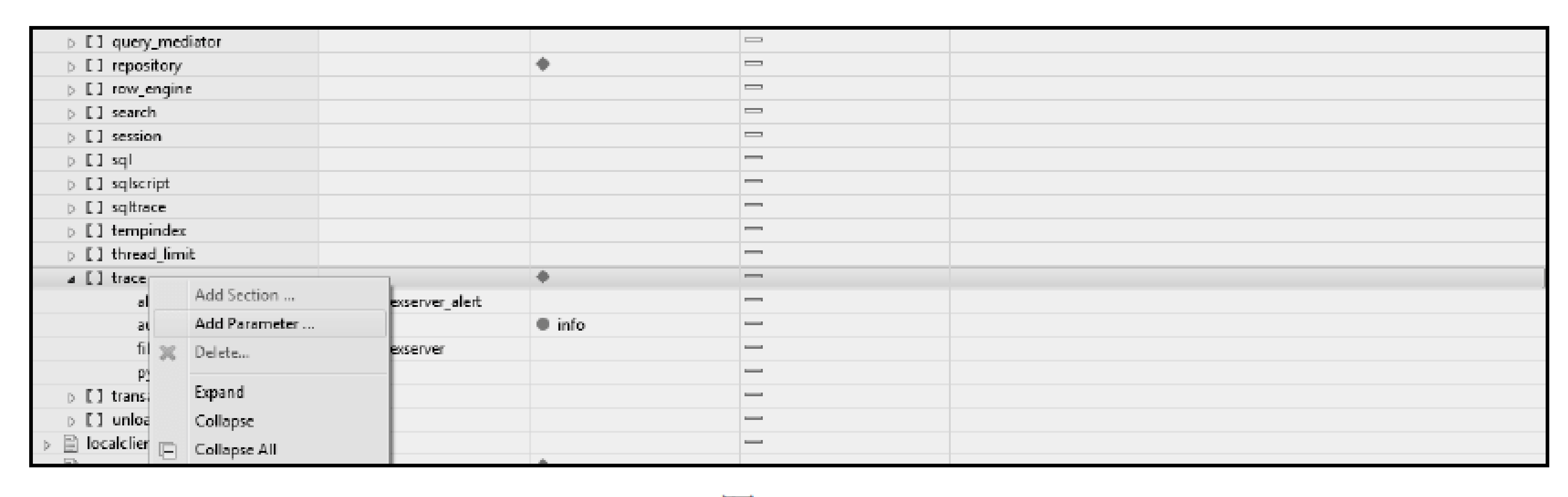
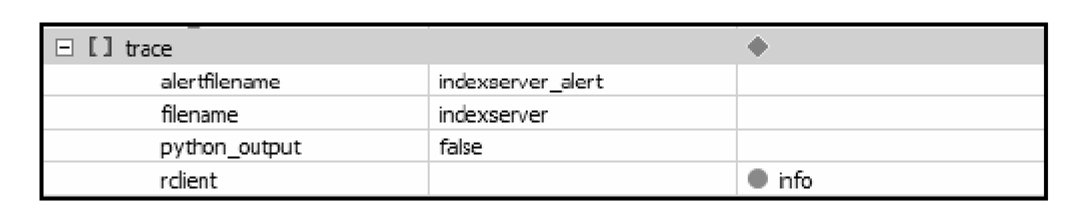


图 13-23

(6) 添加如下参数值，这样我们就开启了 R 的追踪(见图 13-24)：

- KEY: rclient
- VALUE: info



trace		
alertfilename	indexserver_alert	
filename	indexserver	
python_output	false	
rclient		info

图 13-24

(7) 追踪可以在如下日志文件中看到“Administration”→“Diagnosis Files”→

“indexserver_*.trc”（见图 13-25）。

```
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.788690 i RClient        RClient.cpp(01102) : Prepare dataframe sending to
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.788707 i RClient        RClient.cpp(01130) : assign input1 as 'input1'.
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.827423 i RClient        RClient.cpp(01146) : Sending a dataframe to R take
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.827446 i RClient        RClient.cpp(01325) : RClient::transfer data to R:
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.827449 i RClient        RClient.cpp(01326) : whole data transfer to R take
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.827467 i RClient        RserveConnection.cpp(00352) : R-script execution:
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.867884 i RClient        RserveConnection.cpp(00383) : R-script execution:
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.867892 i RClient        RserveConnection.cpp(00384) : R-script execution t
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.867900 i RClient        RClient.cpp(01468) : RClient::transfer data from R
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.867909 i RClient        RserveConnection.cpp(00457) : get result of name '
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.908008 i RClient        RClient.cpp(01924) : RClient::transfer data from R
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.908016 i RClient        RClient.cpp(01925) : whole datatransfer from R tak
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.908048 i RClient        RClient.cpp(01995) : RClient::execute: stop.
[47698]{-1}[-1/-1] 2013-09-18 11:43:44.908052 i RClient        RClient.cpp(01996) : whole R-Op takes 274 ms.
```

图 13-25

第五节　示　　例

在本章的前几节中，我们已经大致了解了 R 语言的基本知识。接下来的例子中，我们会讲述如何通过 R 扩展来迅速实现 SAP HANA 中暂不具备的一些数据分析。首先我们需要 Kernlab 程序包中的 SVM 算法来实现。

一、安装 Kernlab 程序包

在 R 服务器，先安装另外一个包 Kernlab。你可以从 CRAN 官网或者此处下载 http://mirrors.softliste.de/cran/web/packages/kernlab/index.html。下载完成后，在控制台中输入如下命令:

```
R
install.packages("/PATH/TO/YOUR/kernlab.tar.gz", repos = NULL)
library("kernlab") #如果装载成功，则不会产生任何信息
q()
```



二、准备数据

其次，我们必须先准备好相应的数据与数据结构。在 SAP HANA Studio 中调出 SQL 控制台，并执行如下 SQL 语句:

```
DROP TABLE IRIS;
CREATE INSERT ONLY COLUMN TABLE IRIS
(ID INTEGER, sepal_length DOUBLE, sepal_width DOUBLE,
```

```
petal_length DOUBLE,  petal_width DOUBLE,  species VARCHAR(10));

DROP TYPE iris_data_type;
CREATE TYPE iris_data_type AS table
(ID INTEGER,  SEPAL_LENGTH DOUBLE,  SEPAL_WIDTH DOUBLE,
PETAL_LENGTH DOUBLE,  PETAL_WIDTH DOUBLE,  SPECIES VARCHAR(10));

DROP PROCEDURE get_iris_data;
CREATE PROCEDURE get_iris_data(OUT res iris_data_type)
LANGUAGE RLANG AS
BEGIN
library(kernlab)
data(iris)
res <- data.frame(cbind(seq(1,  150,  1),  iris))
colnames(res) <- c("ID",  "SEPAL_LENGTH",  "SEPAL_WIDTH",  "PETAL
_LENGTH",  "PETAL_WIDTH",  "SPECIES")
END;

CALL get_iris_data(IRIS) WITH OVERVIEW;
```

我们可以通过语句 select*from IRIS 来查看所有 IRIS 表中的数据。不过，ISIS 表并不是我们想要处理的最终表，我们会继续创建拥有更多属性的另外一个表：

```
DROP TABLE IRIS_GENERIC;
CREATE  INSERT  ONLY  COLUMN  TABLE  IRIS_GENERIC  (sepal_length
DOUBLE,  sepal_width DOUBLE,  petal_length DOUBLE,  petal_width
DOUBLE,
ctraining INTEGER,  clabel VARCHAR(10),  cpredict VARCHAR(10),
cprob DOUBLE);

INSERT INTO IRIS_GENERIC (sepal_length,  sepal_width,  petal_len
gth,  petal_width,  ctraining,  clabel,  cpredict,  cprob)
    SELECT sepal_length,  sepal_width,  petal_length,  petal_width,
    1,  species,  'NA',  -1.0  FROM IRIS;
```

同样，可以在 SAP HANA Studio 中检查表 IRIS_GENERIC 中的数据。

三、创建 R 连接表

接下来，我们再额外创建一个连接表来储存 SAP HANA 服务器信息供 R 调用。注意这里的信息应该根据你自己的服务器情况填写：

```
DROP TABLE HDB_CONNECTIVITY;
```

```
CREATE  TABLE  HDB_CONNECTIVITY  (CONN_TYPE  VARCHAR(50) ,   ATTR
VARCHAR(50),  VAL VARCHAR(200));

INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'DSN',  'hana');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'DRIVER',  'HDBODBC');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'UID',  'system');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'PWD',  'manager');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'SERVERDB',  'XXX');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'HOST',  'localhost');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'PORT',  '30015');
INSERT INTO HDB_CONNECTIVITY VALUES ('ODBC',  'DATABASE',  'SYSTEM');

INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'SID',  'HDB');
INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'INSTANCE',  '03');
INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'PORT',  '30315');
INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'HOST',  'localhost');
INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'UID',  'SYSTEM');
INSERT INTO HDB_CONNECTIVITY VALUES ('JDBC',  'PWD',  'MANAGER');

DROP TYPE HDB_CONNECTIVITY_T;
CREATE TYPE HDB_CONNECTIVITY_T AS TABLE (CONN_TYPE VARCHAR(50),
ATTR VARCHAR(50),  VAL VARCHAR(200));
```

四、训练数据

在数据分析的过程中，经常我们会把它分成两个步骤：第一步使用训练数据得到数据规则，第二步使用预测数据来预测相应的结果。在这里我们也同样地分成两步：



- 继续创建数据类型：

```
DROP TYPE iris_gen_type;
CREATE TYPE iris_gen_type AS TABLE ( SEPAL_LENGTH DOUBLE,
                                     SEPAL_WIDTH DOUBLE,
                                     PETAL_LENGTH DOUBLE,
                                     PETAL_WIDTH DOUBLE,
                                     CTRAINING INTEGER,
                                     CLABEL VARCHAR(32),
                                     CPREDICT VARCHAR(32),
                                     CPROB DOUBLE);
DROP TYPE model_type;
CREATE TYPE model_type AS TABLE(model_id varchar(32));
```

- 创建训练算法：

```
DROP PROCEDURE iris_training;
CREATE  PROCEDURE  iris_training(IN  input1  iris_gen_type ,   IN
input2 HDB_CONNECTIVITY_T, OUT output1 model_type)
    LANGUAGE RLANG AS
    BEGIN
       library("kernlab");
       library("ron");
       library("RODBC");
       input_training <- input1[input1$CTRAINING == 1, ];
       meta_cols <- c("CPREDICT", "CTRAINING",  "CPROB");

       training_features   <-   input_training[-match(meta_cols ,
                     names(input_training))];

model <- ksvm(CLABEL ~ ., data = training_features, type = "C-
bsvc",  kernel  = "rbfdot",  kpar  = list(sigma  = 0.1),  C  = 10,
prob.model = TRUE);
       MODEL_ID = c("IRIS_MODEL");

ODBC_DSN <- input2[which(input2$CONN_TYPE == "ODBC" & input2$ATTR
== "DSN"), 3];
ODBC_DRIVER <- input2[which(input2$CONN_TYPE == "ODBC" & input2
$ATTR == "DRIVER"), 3];
ODBC_UID <- input2[which(input2$CONN_TYPE == "ODBC" & input2$ATTR
== "UID"), 3];
ODBC_PWD <- input2[which(input2$CONN_TYPE == "ODBC" & input2$ATT
R == "PWD"), 3];
ODBC_SERVERDB <- input2[which(input2$CONN_TYPE == "ODBC" & input
2$ATTR == "SERVERDB"), 3];
ODBC_HOST <- input2[which(input2$CONN_TYPE == "ODBC" & input2$A
TTR == "HOST"), 3];

ODBC_PORT <- input2[which(input2$CONN_TYPE == "ODBC" & input2$A
TTR == PORT"), 3];
ODBC_DATABASE <- input2[which(input2$CONN_TYPE == "ODBC" & input
2$ATTR == "DATABASE"), 3];

conn <- odbcDriverConnect(paste("DSN=", ODBC_DSN, ";DRIVER=",
ODBC_DRIVER, ";UID=", ODBC_UID, ";PWD=", ODBC_PWD,
                                ";SERVERDB=", ODBC_SERVERDB, ";
SERVERNODE=", ODBC_HOST, ":", ODBC_PORT, ";DATABASE=", ODBC_
```

```
DATABASE, sep=""), believeNRows = FALSE);

      saveRobject(conn, model, "irisTest", forceUpdate=TRUE);

      output1 <- data.frame(MODEL_ID);
END;
```

- 调用训练算法：

```
CALL iris_training(iris_generic, HDB_CONNECTIVITY, model_name)
 WITH OVERVIEW;
```

五、预测数据

- 创建预测数据类型：

```
DROP TABLE model_name;
CREATE TABLE model_name(MODEL_ID varchar(32));

DROP TABLE iris_predicted;
CREATE TABLE iris_predicted( SEPAL_LENGTH DOUBLE,
                             SEPAL_WIDTH DOUBLE,
                             PETAL_LENGTH DOUBLE,
                             PETAL_WIDTH DOUBLE,
                             CTRAINING INTEGER,
                             CLABEL VARCHAR(32),
                             CPREDICT VARCHAR(32),
                             CPROB DOUBLE);
```



- 预测算法：

```
DROP PROCEDURE iris_prediction;
CREATE PROCEDURE iris_prediction(IN input1 iris_gen_type, IN
input2 model_type, IN input3 HDB_CONNECTIVITY_T, OUT output1
iris_gen_type)
LANGUAGE RLANG AS
BEGIN
     library("kernlab");
     library("ron");
     library("RODBC");

    ODBC_DSN <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "DSN"), 3];
```

```
    ODBC_DRIVER <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "DRIVER"), 3];
    ODBC_UID <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "UID"), 3];
    ODBC_PWD <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "PWD"), 3];
    ODBC_SERVERDB <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "SERVERDB"), 3];
    ODBC_HOST <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "HOST"), 3];
    ODBC_PORT <-  input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "PORT"), 3];
    ODBC_DATABASE <- input3[which(input3$CONN_TYPE == "ODBC" &
    input3$ATTR == "DATABASE"), 3];

    conn <- odbcDriverConnect(paste("DSN=",  ODBC_DSN,
    ";DRIVER=",  ODBC_DRIVER,  ";UID=",  ODBC_UID,  ";PWD=",
    ODBC_PWD,
    ";SERVERDB=",  ODBC_SERVERDB,  ";SERVERNODE=",  ODBC_HOST,
    ":",  ODBC_PORT,  ";DATABASE=",  ODBC_DATABASE,  sep=""),
    believeNRows = FALSE);

     model <- loadRobject(conn,  "irisTest");

     input_test <- input1;
     meta_cols <- c("CPREDICT", "CTRAINING",  "CPROB",
"CLABEL");
     test_features <- input_test[-match(meta_cols,
names(input_test))];

     prob_matrix <- predict(model,  test_features,
type="probabilities");

     clabel <- apply(prob_matrix,  1,  which.max);
     cprob <- apply(prob_matrix,  1,  max);
     classlabels = colnames(prob_matrix);
     clabel <- classlabels[clabel];

     output1 <- input1;
     output1$CPREDICT <- clabel;
     output1$CPROB <- cprob;
END;
```



调用并查看预测结果：

```
CALL iris_prediction(iris_generic, model_name,
HDB_CONNECTIVITY, iris_predicted) WITH OVERVIEW;

SELECT * FROM iris_predicted;
```

这样，我们实现了通过 R Kernlab 程序包中的 KSVM 算法来实现对数据分类对并预测相应数据。

本章小结与练习

在本节开始，我们讲述了 R 语言的一些基本概念与使用：

- R 系统是开源的统计绘图软件，也是一种脚本语言，有大量的程序包可以利用。
- R 系统中的向量、列表、数组、函数等都是对象，可以方便地查询和引用，并进行条件筛选。
- R 语言编写函数无需声明变量的类型，能利用循环、条件语句，控制程序的流程。

然后我们知道了若要将 R 语言与 SAP HANA 集成，我们需要在另外一台 SUSE 服务器安装 Rserve 并在 SAP HANA Studio 中配置参数：

- Cer_timeout：设置连接超时，单位为秒，默认为 300。
- Cer_rserve_addresses：R 主机列表。
- Cer_rserve_maxsendsize：用于表示从 R 语言到 SAP HANA 中传递结果数据大小的最大值。



练习

(1) 安装 R 系统并导入程序包。确认程序包已导入。
(2) 查询 R 系统的帮助文件。
(3) 自己创建一个 CSV 文件，并将其导入至 R 系统中，并使用下标进行批量赋值。
(4) 检查并确认您的 SAP HANA 系统已配置完成 R 语言的连接参数。
(5) 在 SAP HANA 中创建一个 R 存储过程并调用。

第十四章　基于网页环境的 SAP HANA 工具

你有没有想过，假如你需要在客户处调试你的 SAP HANA 应用，或者在客户那里新建演示用途的应用，但是你发现客户的终端没有安装 SAP HANA Studio，在这种情况下你如何来完成这些工作呢？从 SAP HANA SPS06 版本开始，你可以通过基于网页的工具来进行 SAP HANA 应用的开发以及调试工作。这些基于网页的工具使得开发者可以随时随地对 SAP HANA XS 应用进行修改和调试，并且其内置的多项功能可以让初学者更为简单地进行 SAP HANA XS 应用的开发。

目前 SAP HANA 提供的网页开发工具有：

- SAP HANA IDE lite
- SAP HANA Web-based Development Workbench
- SAP HANA XS Administration Tool
- SAP HANA XS Debugging
- SAP HANA Application Lifecycle Manager

第一节　SAP HANA 简化版编辑工具

SAP HANA 简化版编辑工具(SAP HANA IDE lite)从名称所带的“lite”我们就能得知，这个工具是一个用来提供快捷但是相对比较简单功能的开发工具。SAP HAHA 提供这个工具的目的是为了开发者能够不借助于 SAP HANA Studio 就能快速建立 SAP HANA XS 应用，它为开发者提供了非常直观和便捷的开发界面，并且很多操作都可以自动完成，例如“.xsapp”和“.xsaccess”文件会被自动生成。

在进行下面的操作之前，请确保你的 SAP HANA 服务器升级到了 SPS06 版本。SAP HANA IDE lite 支持目前流行的大部分浏览器，例如 Internet Explorer、Mozilla Firefox 和 Google Chrome。但是其内嵌的 Debug 功能只支持 Mozilla Firefox 和 Google Chrome 浏览器。



(1) 请在浏览器中输入 http://<WebServerHost>:80<SAP HANAinstance>/sap/hana/xs/editor 打开此工具(见图 14-1)。

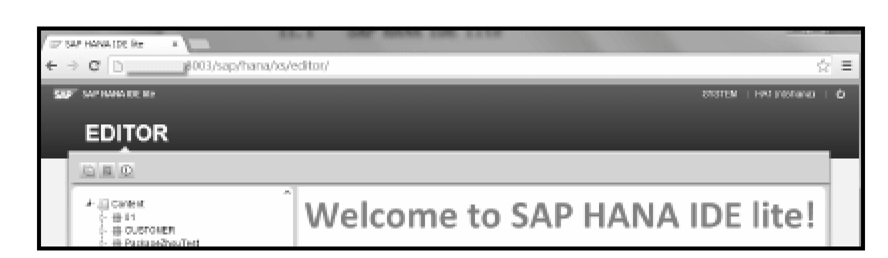


图 14-1

(2) 在开始使用这个工具前，请单击“Information”按钮来了解一些常见问题的解答(见图 14-2)。



图 14-2

打开后的界面如图 14-3 所示。

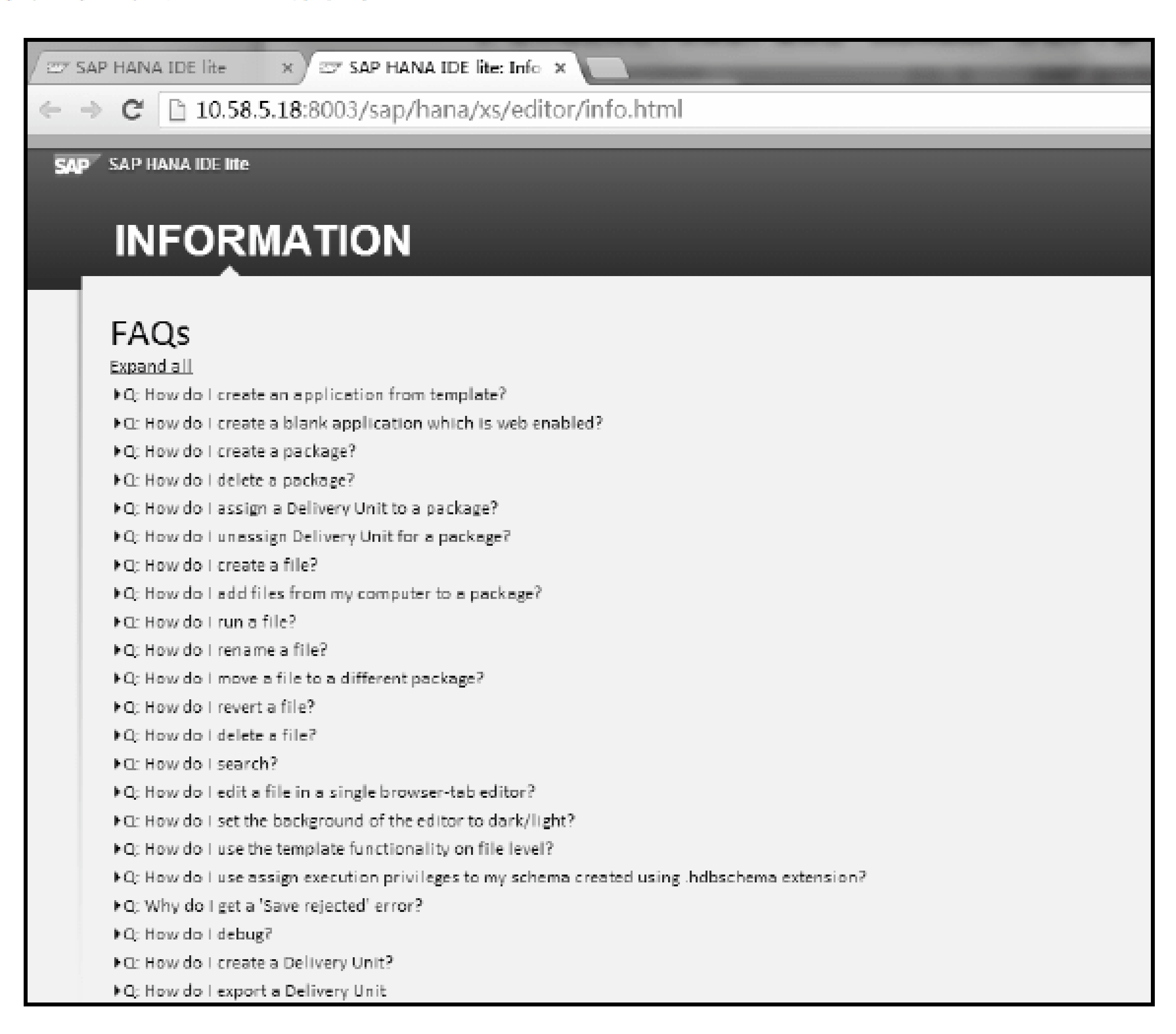


图 14-3



在“INFORMATION”页面里面，你能找到很多常见的问题以及其答案。在你阅读完毕后，我们来看看怎么使用这个工具吧。

一、创建简单的 “Hello World” 应用

还记得我们在第十二章建立的“Hello World”应用吗？这次我们使用 SAP HANA IDE lite 网页工具来看看它是怎么快速建立一个一样的应用的。

(1) 打开 SAP HANA IDE lite。在“Content”节点右击，选择“Create Application”(见图 14-4)。

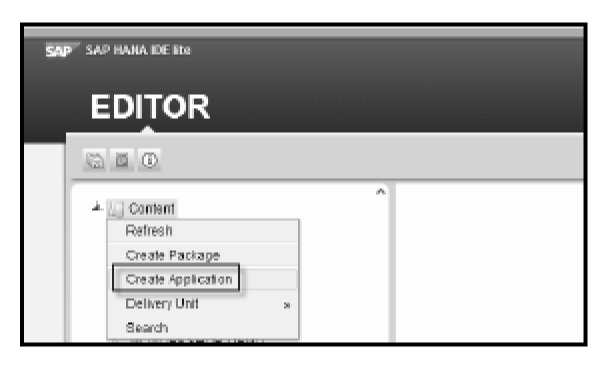


图 14-4

(2) 在弹出窗口中输入所创建应用属于的包。我们在“Template”选项中选择“Hello World”，单击“Create”按钮确认(见图 14-5)。

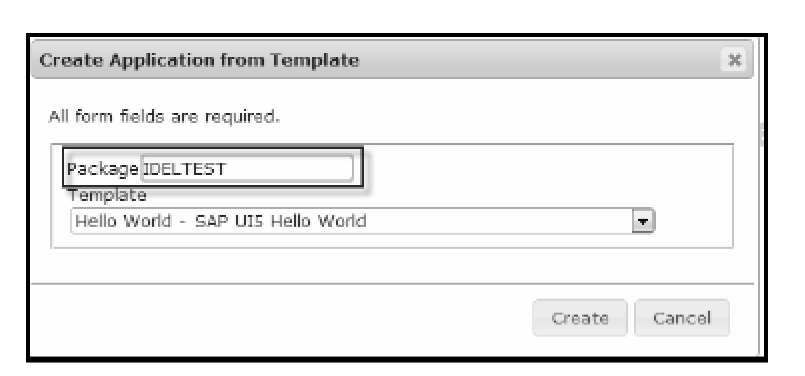


图 14-5

(3) 确认应用在指定的包被成功生成(见图 14-6)。

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>SAPUI5 in 20 Seconds</title>

    <!-- 1.) Load SAPUI5 (from a remote server), select theme and control library -->
    <script id="sap-ui-bootstrap"
        type="text/javascript"
        src="/sap/ui5/1/resources/sap-ui-core.js"
        data-sap-ui-theme="sap_goldreflection"
        data-sap-ui-libs="sap.ui.commons"></script>


    <!-- 2.) Create a UI5 button and place it onto the page -->
    <script>
        // create the button instance
        var myButton = new sap.ui.commons.Button("btn");

        // set properties, e.g. the text (there is also a shorter way of setting several pr
        myButton.setText("Hello World!");

        // attach an action to the button's "press" event (use jQuery to fade out the butto
        myButton.attachPress(function(){$("#btn").fadeOut()});

        // place the button into the HTML element defined below
```

13:41:28 >> Application in Package IDELTEST created successfully.

图 14-6

(4) 在左侧相应的包下，我们能看见系统自动生成了“.xsapp”和“.xsaccess”文件，同时“index.html”文件也被一起生成(见图 14-7)。

(5) 单击“Run”按钮(见图 14-8)。

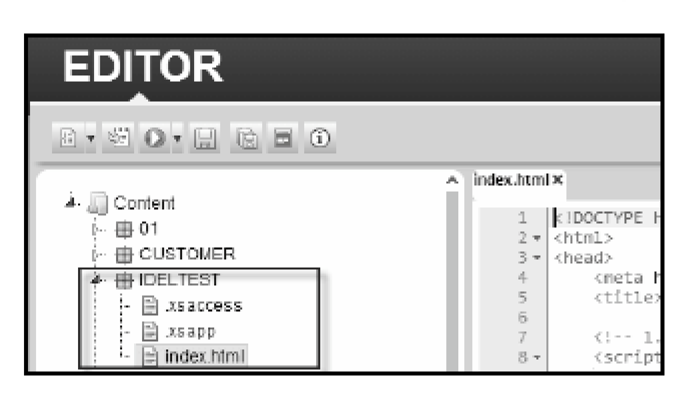


图 14-7



图 14-8

(6) 在新弹出页面检查运行结果。单击“Hello World！”按钮，看看会发生什么吧(见图 14-9)。



图 14-9

至此我们已经完成了一个应用的创建，非常简单吧。在下文中，我们来创建一个不一样的应用。

二、快速为移动设备创建应用



上面我们说要做一个不一样的应用，看到标题你就能知道在哪里不一样了，因为这个应用是为了移动设备创建的。

(1) 打开 SAP HANA IDE lite。在“Content”节点右击，选择“Create Application”(见图 14-10)。

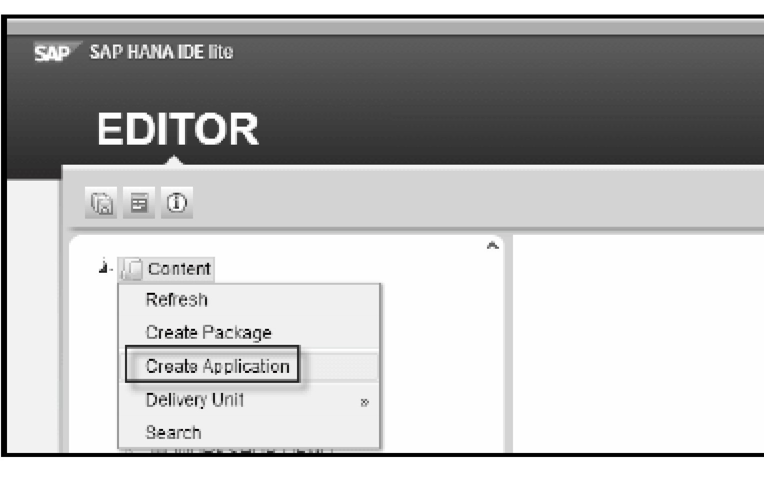


图 14-10

(2) 在弹出窗口中输入所创建应用属于的包。我们在“Template”选项中选择“Simple Mobile Application”，单击“Create”确认(见图 14-11)。

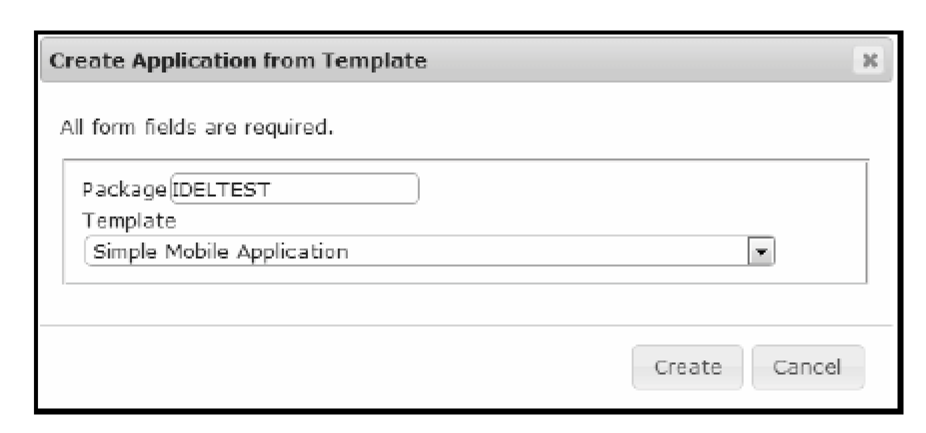


图 14-11

(3) 检查生成的应用(见图 14-12)。

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <title>Mobile App in 23 Seconds Example</title>

    <script src="/sap/ui5/1/resources/sap-ui-core.js"
        id="sap-ui-bootstrap"
        type="text/javascript"
        data-sap-ui-libs="sap.m"
        data-sap-ui-theme="sap_mvi">
    </script>
    <!-- only load the mobile lib "sap.m" and the "sap_mvi" theme -->


    <script>

      // create a mobile App
      // it initializes the HTML page for mobile use and provides animated page handling
      var app = new sap.m.App("myApp", {initialPage:"page1"}); // page1 should be displa


      // create the first page of your application
      var page1 = new sap.m.Page("page1", {
```

14:29:24 >> Application in Package IDELTEST created successfully.

图 14-12

(4) 单击“Run”按钮运行(见图 14-13)。

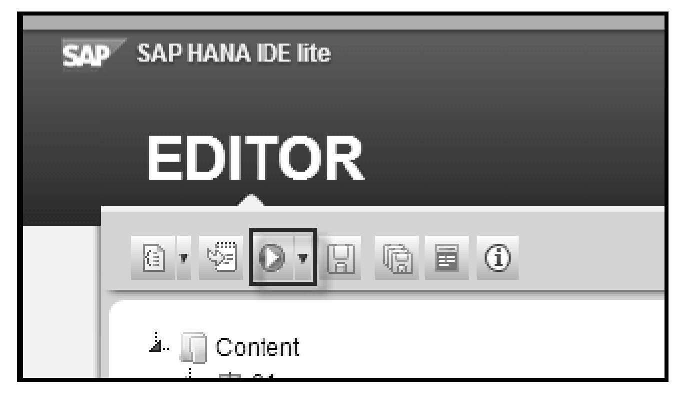


图 14-13

(5) 由于我们创建的是移动设备应用，因此我们在台式计算机上运行时系统会提示错误。如有条件请使用运行 iOS 或者 Android 系统的设备来测试结果(见图 14-14)。





图 14-14

三、创建服务器端 JavaScript 应用

我们在第十二章已经介绍了如何创建 SAP HANA 服务器端应用，在本小节中我们来看看如何使用 SAP HANA DIE lite 工具来轻松创建 SAP HANA 服务器端应用。

(1) 打开 SAP HANA IDE lite。在“Content”节点右击，选择“Create Application”(见图 14-15)。

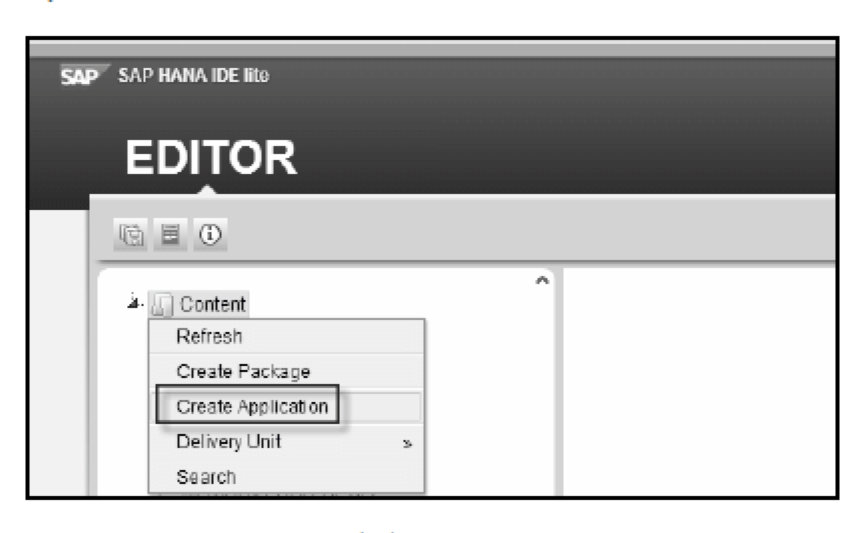


图 14-15

(2) 在弹出窗口中输入所创建应用属于的包。我们在“Template”选项中选择“Blank Application”，单击“Create”按钮确认(见图 14-16)。

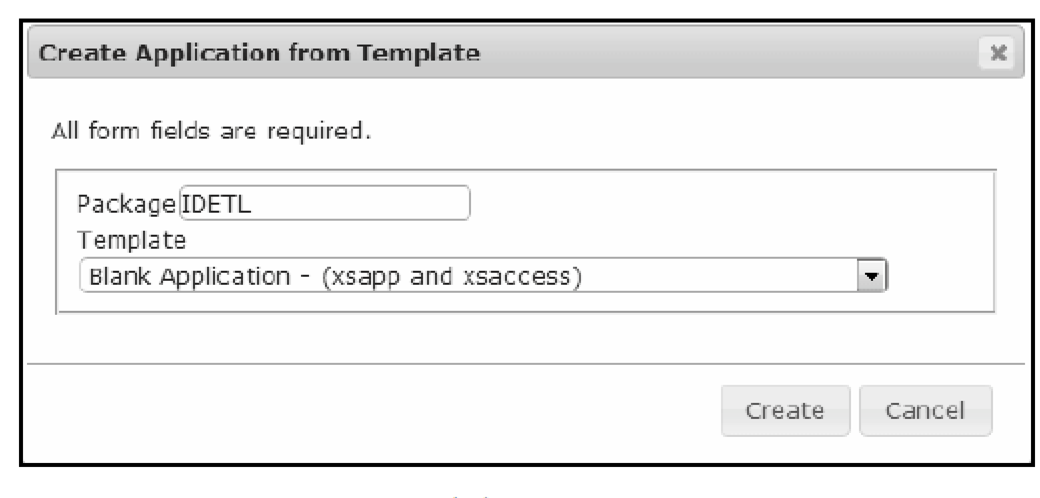


图 14-16



(3) 在“Content”节点下对应的包中，你能找到自动创建的程序文件，请在“index.html”文件上右击鼠标，选择“Delete”操作删除此文件。如果你没有找到对应的包，请刷新你的浏览器(见图 14-17)。

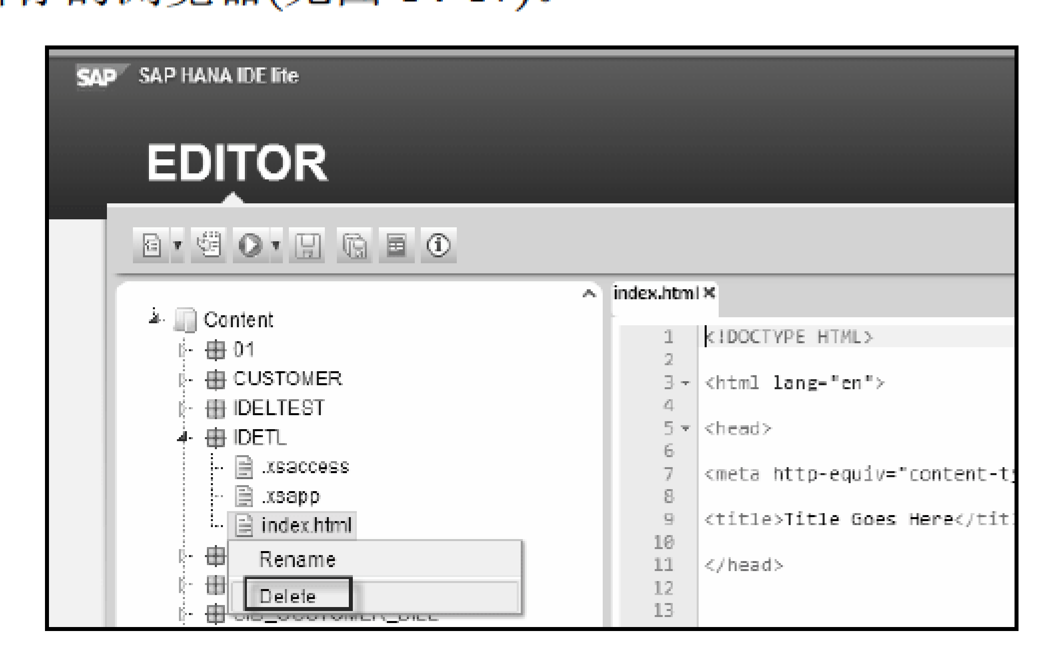


图 14-17

(4) 在新建的包上右击，选择“Create File”(见图 14-18)。

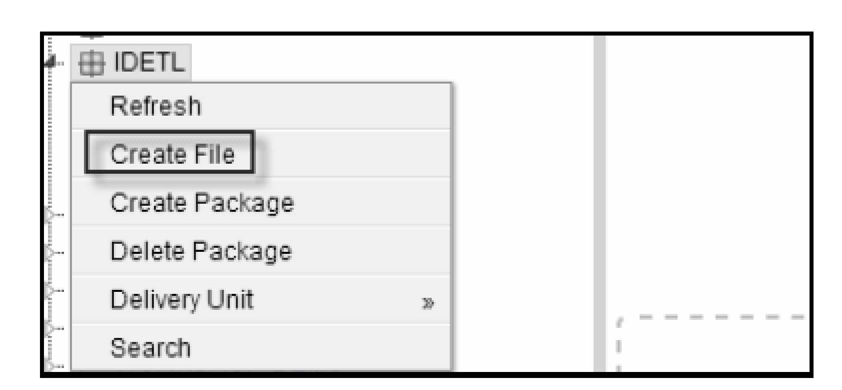


图 14-18

(5) 重命名新生成文件为“MyFirstApp.xsjs”(见图 14-19)。

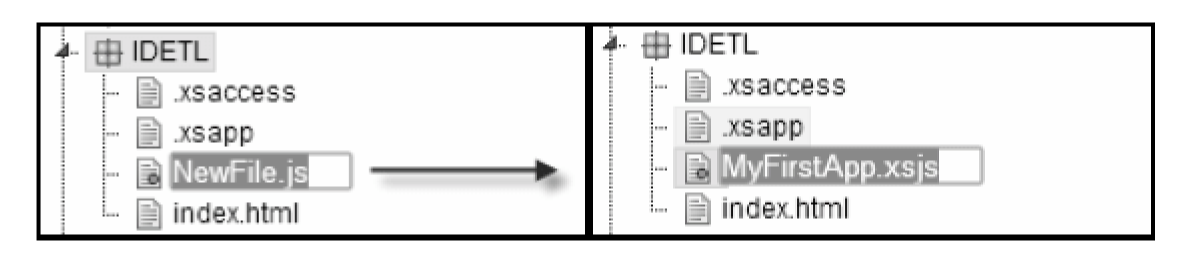


图 14-19

(6) 将如下代码输入到屏幕右侧空白处，如图 14-20 所示。

```
$.response.contentType = "application/json";
$.response.status = 200;
$.response.contentType = "text/plain";

var variable1 = $.request.parameters.get("var1");

var variable2 = $.request.parameters.get("var2");
```




```
try { switch($.request.parameters.get("mode"))

{ case "multiply": $.response.setBody(doMultiply(variable1, variable2)); break;

case "add": $.response.setBody(doAdd(variable1, variable2)); break;

default: $.response.setBody("Service not supported: "+$.request.parameters.get("mode")); break; }}

catch (err) { $.response.setBody("Failed to execute action: "+err.toString());}

function doMultiply(var1, var2){ return var1+" X "+var2+" = "+var1*var2;}

function doAdd(var1, var2){ return var1+" + "+var2+" = "+(parseInt(var1)+parseInt(var2));}
```

*MyFirstApp.xsjs

```
$.response.contentType = "application/json";

$.response.status = 200;

$.response.contentType = "text/plain";

var variable1 = $.request.parameters.get("var1");

var variable2 = $.request.parameters.get("var2");

try { switch($.request.parameters.get("mode"))

{ case "multiply": $.response.setBody(doMultiply(variable1, variable2)); break;

case "add": $.response.setBody(doAdd(variable1, variable2)); break;

default: $.response.setBody("Service not supported: "+$.request.parameters.get("mode")); break; }}

catch (err) { $.response.setBody("Failed to execute action: "+err.toString());}

function doMultiply(var1, var2){ return var1+" X "+var2+" = "+var1*var2;}

function doAdd(var1, var2){ return var1+" + "+var2+" = "+(parseInt(var1)+parseInt(var2));}
```

图 14-20



和第十二章一样，我们在这里也是生成一个计算类应用，即通过判断参数“mode”来执行相应的运算操作，而参数“var1”和“var2”则负责传递具体的需要计算的数值。在这里我们需要着重提一下 SAP HANA DIE lite 工具的强大之处，我们知道通常我们在 SAP HANA Studio 中要使得 XSJS 应用能够运行，需要在“Project Explorer”视图中提交并激活应用，然后才能在浏览器中调用它，在 SAP HANA DIE lite 工具中，一个简单的“Save”按钮就帮你解决了上述操作。

(7) 单击“Save”按钮激活应用。请注意在屏幕下方提示框里面的信息，应用在

保存的同时已经被激活(见图 14-21)。

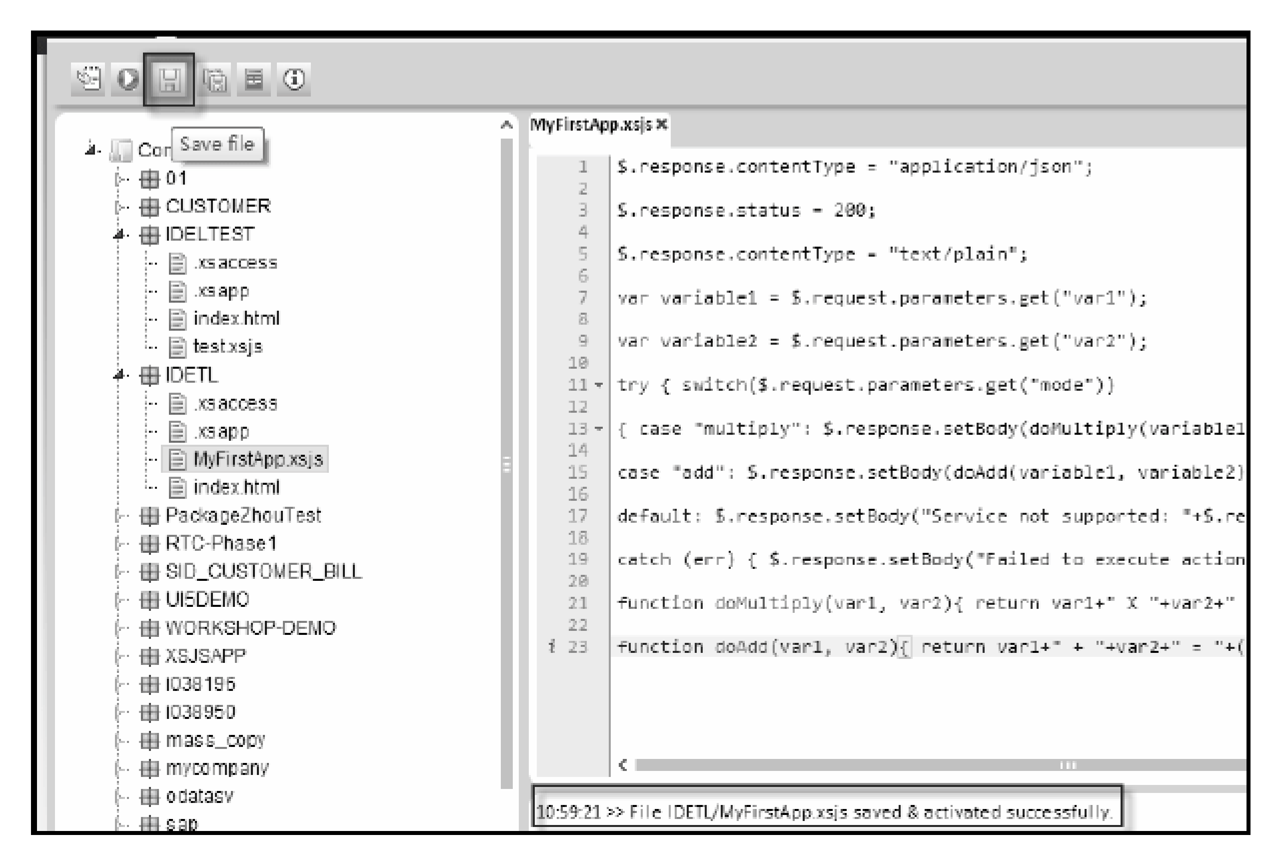


图 14-21

(8) 单击“Run”按钮在浏览器中调用应用(见图 11-22)。

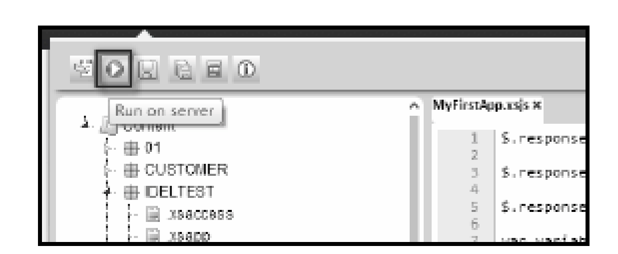


图 11-22

(9) 由于初始时我们没有传递任何参数，所以系统提示如图 14-23 所示的信息。



图 14-23

在此步中，我们将 URL 中的参数加上：“<your_file_name>.xsjs?mode=multiply &var1=10&var2=200”。这段参数表明我们需要乘法运算，乘数是“200”，被乘数是“10”。

(10) 将修改好的 URL 再次输入到浏览器中并调用(见图 14-24)。



图 14-24



四、Debug 服务器端 JavaScript 应用

到目前为止，我们已经知道了如何利用 SAP HANA DIE lite 工具来快速创建 SAP HANA XSJS 应用，那么对于我们日常的开发工作来讲，程序出现问题是经常出现的问题，要解决问题我们通常会使用“Debug”的方法来发现问题究竟出现在什么地方。现在我们介绍一下使用 SAP HANA DIE lite 工具来“Debug”SAP HANA XSJS 应用的方法。

在继续下面的步骤之前，请将“sap.hana.xs.debugger::Debugger”角色赋予操作用户。

(1) 我们在上一小节的例子中将传递参数的 URL 改成“<xxx>.xsjs?mode=minus-&var1=10&var2=200”，即我们让这个应用做减法，如图 14-25 所示。



图 14-25

从上图我们能看出，应用返回了“减法不被支持”的信息，那么这条信息是从哪里发出的呢，我们来通过“Debug”方式找到相应的代码吧。

(2) 返回应用编写界面，在第 11 行处点击行代码(图 14-26 中框标出处)。

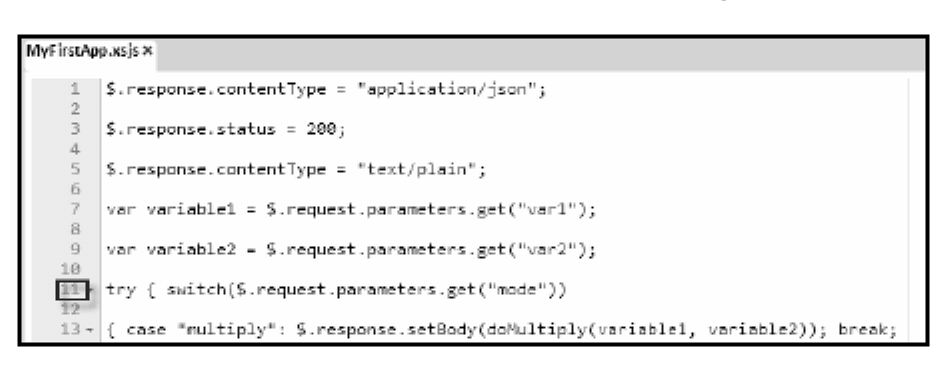


图 14-26

我们之所以选择第 11 行，是因为前面的语句都是定义 API 和变量的语句，与程序的运算无关，只有从第 11 句开始，程序会做一些逻辑的判断以及执行相应的操作。

单击后我们能看到第 11 行被打上了“断点”标签。



MyFirstApp.xsjs

```
1  $.response.contentType = "application/json";
2
3  $.response.status = 200;
4
5  $.response.contentType = "text/plain";
6
7  var variable1 = $.request.parameters.get("var1");
8
9  var variable2 = $.request.parameters.get("var2");
10
11 try { switch($.request.parameters.get("mode"))
12
13 { case "multiply": $.response.setBody(doMultiply(variable1, variable2)); break;
14
15 case "add": $.response.setBody(doAdd(variable1, variable2)); break;
16
17 default: $.response.setBody("Service not supported: "+$.request.parameters.get("mode")); break; }}
18
19 catch (err) { $.response.setBody("Failed to execute action: "+err.toString());}
20
21 function doMultiply(var1, var2){ return var1+" X "+var2+" = "+var1*var2;}
22
23 function doAdd(var1, var2){ return var1+" + "+var2+" = "+(parseInt(var1)+parseInt(var2));}
```

图 14-27

(3) 再次在浏览器中调用此应用后，会发现页面会一直停在加载状态，并且页面的状态栏会提示等待服务器的响应(见图 14-28)。



图 14-28

(4) 接着我们再返回程序编辑界面(见图 14-29)。

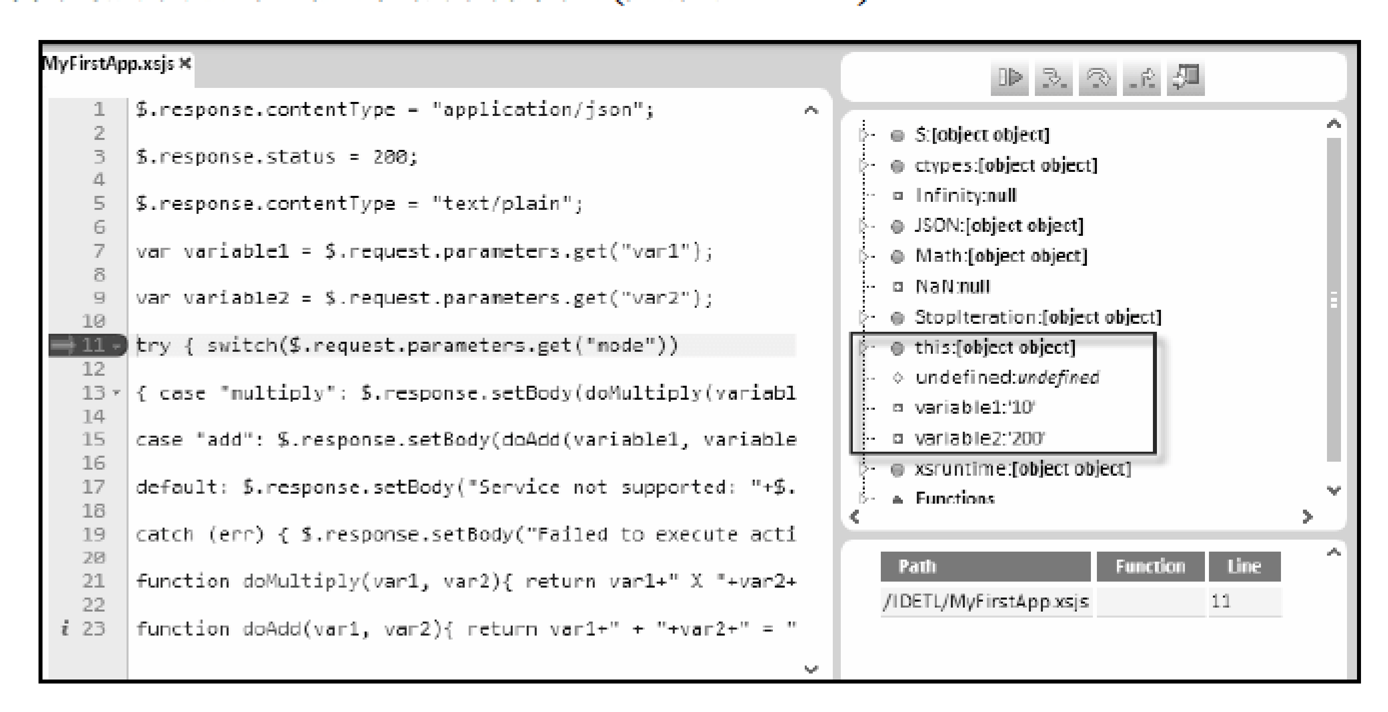


图 14-29

从上图可以看出，程序运行停在了第 11 行，从右侧的调试信息框能看出，参与计算的变量(“variable1”和“variable2”)已经被赋值，但是负责计算方法的变量(“mode”)为空。请注意调试信息框上方的 5 个按钮，它们在“Debug”过程中有

很大的用途，从左到右这 5 个按钮的作用为：

- Resume：恢复；
- Step in：单步进入；
- Step over：单步跳过；
- Step out：单步跳出；
- Change to development perspective：关闭调试，返回编辑视图。

经过逐步调试，我们能发现在第 17 行的语句生成了在本小节最开始返回的信息，并且我们还能进一步发现该应用只能进行加法和乘法的运算，不能进行减法的运算，所以才会出现服务不支持的提示。随着今后我们开发的应用逐渐复杂，我们会越来越多的用到“Debug”功能来调试程序，因此我们推荐大家应该熟练掌握好本小节的内容，并能在今后的开发工作中多加尝试。

第二节　其他工具概览

在介绍完 SAP HANA DIE lite 这个网页工具后，细心的读者可能对其他的 SAP HANA 提供的网页工具也非常感兴趣，但是受篇幅所限我们在这里只列出其他工具的用途以及如何通过浏览器来访问这些工具。在这里我们还是要再提及一下 SAP Community Network(http://scn.sap.com/welcome)这个网站，因为几乎所有的工具你都能在这里找到相应的案例讲解和问题答复，有时间的话，多来看看吧。

接下来我们以知识点的方式来介绍下其他的网页工具：

- SAP HANA XS Administration Tool：主要完成与应用基本设置相关的工作，包括应用安全配置、SAML 配置、“HTTP destinations”配置以及信任管理等。我们在浏览器调用此工具的方式为“http://<WebServerHost>:80<SAP-HANAinstance>/sap/hana/xs/admin”(见图 14-30)。

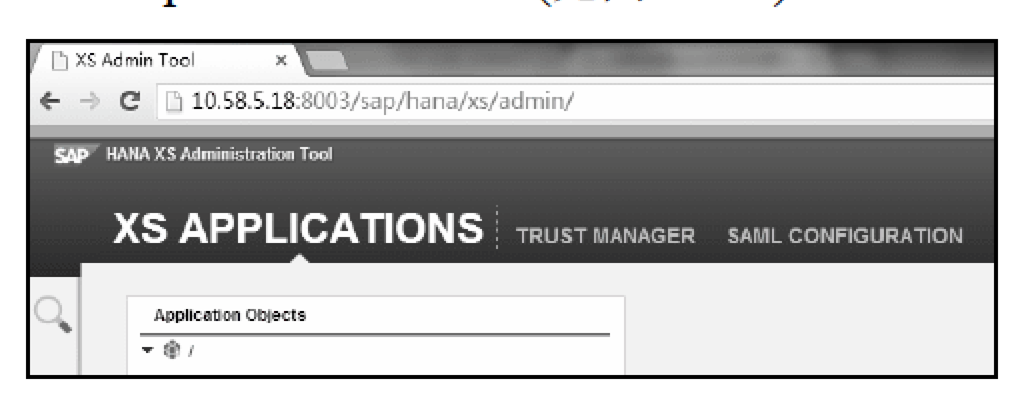


图 14-30

- SAP HANA XS Debugging：为其他用户赋予 Debug 进程权限。我们在浏览器调用此工具的方式为“http://<WebServerHost>:80<SAP HANAinstance>/sap/hana/xs/debugger”(见图 14-31)。
- SAP HANA Web-based Development Workbench：这个工具是我们在上一小节介绍的 SAP HANA DIE lite 工具的完整版。这个工具涵盖了绝大部分 SAP HANA Studio 的功能，你可以使用这个工具完成创建交付单元，提交和激活 SAP HANA 项目文件，创建 Schema，创建数据表等工作，你甚至还可以使用这个工具来修改已经在 SAP HANA 服务器中存在的各种视图。我们在浏览器调用此工具的方式为“http://<WebServerHost>:80<SAP HANAinstan-ce>/sap/hana/xs/ide”(见图 14-32)。

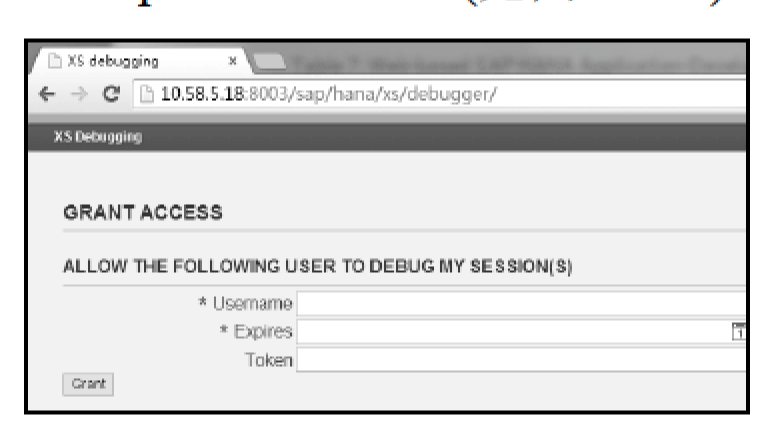


图 14-31

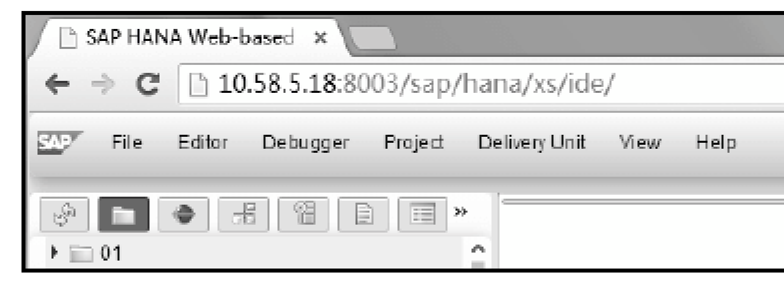


图 14-32

- SAP HANA Application Lifecycle Manager：用来管理完整的产品生命周期。我们在浏览器调用此工具的方式为“http://<WebServerHost>:80<SAPHAN-Ainstance>/sap/hana/xs/lm”(见图 14-33)。

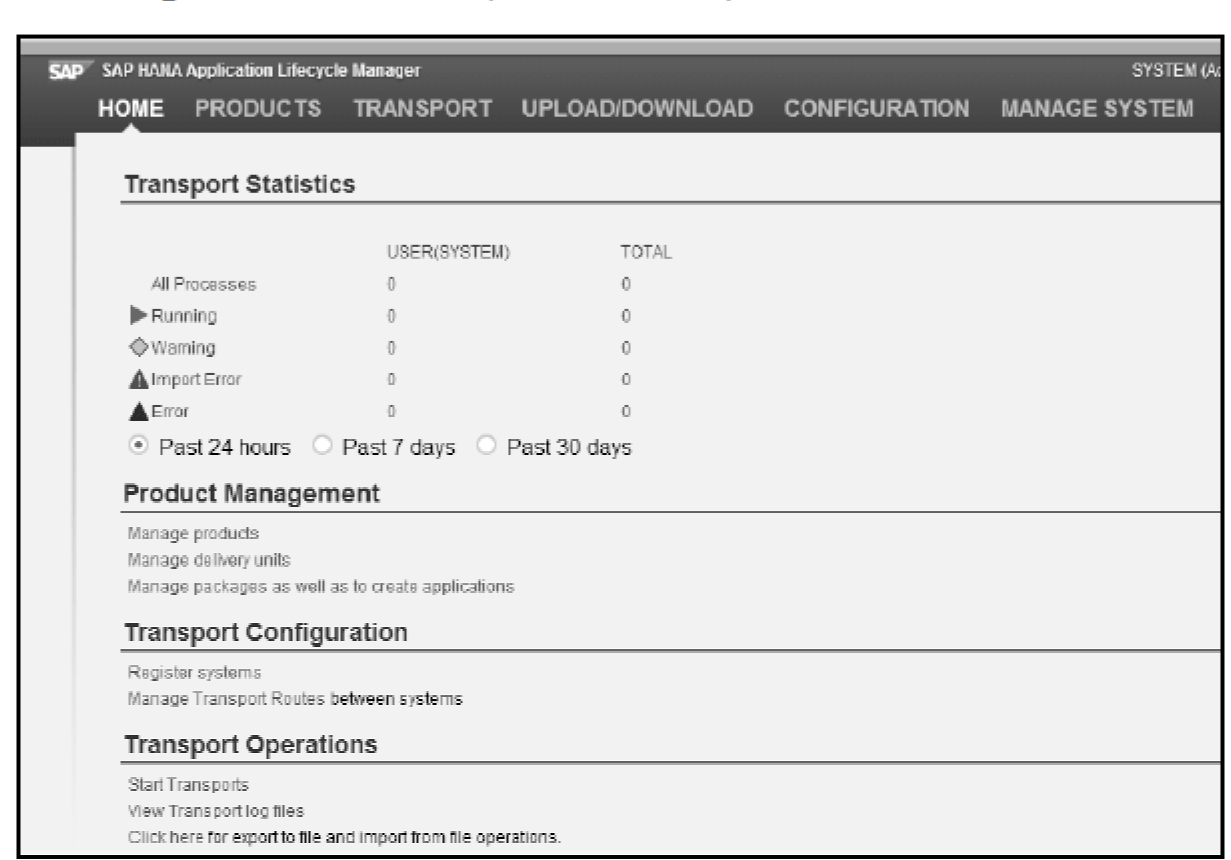


图 14-33



本章小结与练习

我们在本章中介绍了如何用 SAP HANA SPS06 版本新发布的网页工具 SAP HANA DIE lite 来快速生成各种应用以及如何调试应用的方法，与此同时我们还以知识点的方式概述了与 SAP HANA DIE lite 工具同时发布的很多非常有用的工具。通过熟悉和使用这些工具，我们希望大家能总结出每种工具的自身特点，并且结合实际应用来不断提高开发的效率。

练习

1. 使用 SAP HANA DIE lite 工具创建一个“Hello World”应用。
2. 使用 SAP HANA DIE lite 工具创建一个移动设备应用，并尝试用移动设备访问测试。
3. 使用 SAP HANA Web-based Development Workbench 工具创建一个交付单元和一个包。
4. 使用 SAP HANA Web-based Development Workbench 工具创建一个项目，并尝试建立 OData 服务。

